Sistemas Informáticos

Curso 2010 – 2011

Entorno virtual 3D multiusuario para simulación de escenarios de evacuación

Fernando Cruz Ruiz

Alberto Durbey Carrasco

Jorge Sanz Briones

Dirigido por:

Pablo Gervás Gómez-Navarro

Carlos León Aznar

Facultad de Informática

Universidad Complutense de Madrid

Entorno virtual 3D multiusuario para Simulación de escenarios de Evacuación

Proyecto de Sistemas Informáticos
Facultad de Informática

Universidad Complutense de Madrid

Autores:
Fernando Cruz Ruiz
Alberto Durbey Carrasco
Jorge Sanz Briones

Profesores directores:
Dr. Pablo Gervás Gómez-Navarro
Dr. Carlos León Aznar

Curso 2010 / 2011

Agradecimientos

Fernando: A mis padres y hermana por el apoyo durante todos estos años. A mi novia por aguantarme este año. A mis amigos que llevan toda la vida a mi lado, a mis cierrabares por este gran año. A Paco y a Iván. Sin su apoyo esto habría no habría sido posible

Alberto: Agradezco a mi familia el apoyo prestado a lo largo de estos años, en especial a mis padres y a mi hermano. A mi novia y amigos por los ánimos en momentos difíciles.

Jorge: A mi familia, a mis amigos y a mi novia Jara, por su apoyo a lo largo de este año tan duro y durante toda la carrera.

También queríamos dar las gracias a los Drs. Carlos León, Gonzalo Méndez y Pablo Gervás que nos han sabido guiar siempre que ha sido necesario.

Fernando Cruz Ruiz, Alberto Durbey Carrasco y Jorge Sanz Briones

RESUMEN

El uso de entornos virtuales 3D en el ámbito de la simulación de protocolos de evacuación permite avanzar en el desarrollo de estos sistemas, mejorando su efectividad y reduciendo los costes. Este documento recoge el estudio previo y la posterior implementación de un entorno virtual 3D que servirá para simular incendios en la Facultad de Informática de la Universidad Complutense de Madrid. Es un entorno multiusuario cliente – servidor.

El documento plantea en primer lugar las limitaciones que presentan actualmente los simulacros de protocolos de evacuación. Se explican posteriormente tanto los orígenes y la evolución de los entornos virtuales, como la integración de los mismos en este tipo de actividades.

Finalmente se expone la evolución de las soluciones aportadas para la consecución de los objetivos y las posibles aplicaciones y ampliaciones de este trabajo.

ABSTRACT

The use of 3D virtual environments in the field of evacuation simulations enables the progress in the development of these systems, improving their effectiveness and reducing their costs. This document depicts the prior study and later implementation of a 3D virtual environment that may be used for fire simulations in the Facultad de Informática (IT School) of the Universidad Complutense de Madrid. It is a multi user client/server environment.

The document sets forth the limitations currently entailed in evacuation simulations, followed by the explanation of the origins and evolution of virtual environments and their integration in this type of activity.

Finally, the evolution of the proposed solutions for the completion of goals is presented, as well as the possible applications and extensions of this project.

Palabras clave para búsquedas bibliográficas

- *Entorno Virtual*
- *Realidad Aumentada*
- *Sistemas multiusuario*
- *Simuladores de Evacuación*
- *Motor Gráfico*

Índice

Capítulo1:
Introducción

1 Introducción

1.1 Sobre el proyecto

Realizar un simulacro de evacuación en la vida real es algo muy costoso y pesado. Hay que movilizar al personal, explicar los prototipitos, y los resultados obtenidos son muy difíciles de cuantificar. Debido a estos problemas surge este proyecto. Se trata de un entorno virtual donde practicar y probar protocolos de evacuación en situaciones de peligro. Mediante la comunicación entre los usuarios que harán de afectados y un guía que le proporcione indicaciones hacia una ruta de escape libre de riesgo, se obtendrá la información necesaria para la creación de un lenguaje natural acorde a estas situaciones. Todas estas conversaciones se recogerán en un log para su posterior análisis. Además, se podrá utilizar la interfaz desarrollada para poder guiar a los afectados de una forma automática.

Este proyecto forma parte del proyecto denominado “NOVA, Navegación basada en Ontologías mediante Verbalización de mensajes de Ayuda ". Se trata de un proyecto encuadrado en el "Plan Nacional de I+D+i del Ministerio de Ciencia e Innovación". El investigador principal es Pablo Gervás Gómez-Navarro y es un proyecto en el que participan las universidades UCM, UPM y la Universidad de Sevilla, con un total de nueve integrantes. Este proyecto se viene desarrollando desde el año 2010 y su finalización se estima para el año 2012, son prácticamente 3 años de investigación y desarrollo del mismo.

El papel que juega el proyecto desarrollado dentro del proyecto “NOVA” es el de proporcionar los elementos necesarios para el desarrollo de un entorno virtual donde poder poner en práctica simulaciones de situaciones de riesgo, así como de los protocolos de gestión en dichas situaciones. El objetivo del proyecto “NOVA” es el de poder orientar a una persona dentro de un edificio mediante indicaciones. Estudia las diferentes formas de comunicación de la gente, así como la eficiencia de las indicaciones dadas.

Todo esto se desarrolla en un entorno virtual del modelado de la facultad de Informática de la universidad Complutense de Madrid. Es este entorno en donde se desarrollaran las simulaciones y los protocolos que se deseen probar. Mediante el uso de un avatar, el usuario podrá recorrer libremente el entorno desarrollado y en caso de necesitarlo, pedir ayuda para encontrar la ruta de escape con menor riesgo.

Combinando lo anteriormente citado y haciendo uso de tecnologías como la realidad aumentada o el posicionamiento en interiores a través de la red inalámbrica Wifi mediante el uso de cualquier dispositivo móvil, se podría llevar a cabo la evacuación de personas en situaciones de peligro siguiendo protocolos ensayados de forma informatizada.

La investigación del proyecto nacional “NOVA” se centra en el uso del lenguaje específico que utilizan las personas en las situaciones de peligro que se quieren evaluar. Mediante el uso del entorno desarrollado para la comunicación entre las personas guiadas y la persona que guía, se buscarán patrones en dicha de comunicación. Se analizarán las conversaciones mantenidas entre grupos de usuarios de estos entornos con el fin de comprender cuales son las frases más comunes, cómo es la forma de pedir una indicación o si es eficiente la indicación dada para obtener el resultado esperado (León, de la Puente, Dionne, & Gervás, 2009).

Una vez realizado el trabajo de campo el resultado final sería una automatización de la comunicación entre las personas afectadas en una situación de riesgo y un guía no físico. Este guía sería capaz de conducir a los afectados hasta una vía de escape en el caso de una situación de peligro o hasta el lugar indicado por el usuario en el caso de pedir las indicaciones para dirigirse a algún lugar del que desconoce su ubicación.

1.2 Presentación del problema

Un incendio es un conjunto de situaciones aleatorias donde el número de variables a tener en cuenta es muy elevado. Recrear estos sucesos en la vida real es algo que difícil y bastante costoso. El problema que el proyecto trata de resolver es el de guiar a una o varias personas en una situación de incendio dentro de un edificio.

En estas situaciones deben tenerse en cuenta muchas posibilidades, la primera y principal es el fuego, dependiendo de donde se origine el incendio y de lo que use como combustible se pueden clasificar los tipos de fuegos en cinco clases, según la normativa Europea:

- Clase A: Son los fuegos más comunes, son aquellos provocados por materiales sólidos, los cuales son casi siempre de naturaleza orgánica. Su combustión se realiza normalmente con la formación de brasas. Este tipo de fuego es el que nos podríamos encontrar en un edificio cerrado.
- Clase B: Son aquellos fuegos de líquidos o de sólidos licuables. Dentro de este grupo podríamos destacar los incendios de petróleo o gasolina, algunas ceras y plásticos.
- Clase C: Incendios que implican gases inflamables, tales como gas natural, gas butano, propano o hidrógeno. Según el tipo de edificio del que se trate, este tipo de fuegos también puede producirse en ellos.
- Clase D: Incendios que implican metales combustibles. Serían. sodio (Na), magnesio (Mg) y potasio (K).
- Clase F: Son los fuegos provocados por la utilización de los derivados de los aceites destinados a la cocina. El problema que este tipo de incendios tiene y por lo que se les separa en un grupo diferente, es porque las altas temperaturas derivadas del incendio de estos aceites hace ineficientes los métodos normales de extinción.

La simulación de todos los factores presentes en un fuego es el gran problema que se presenta en la vida real, pero gracias a entornos virtuales donde se pueden recrear infinidad de situaciones, la preparación previa para afrontar este tipo de contingencias podría ser mejor y más eficiente. Este es el problema que el proyecto desarrollado trata de solucionar, intentando ser lo más fiel posible a situaciones que se pudieran dar en la vida real.

Por otro lado se trata la comunicación. La manera de comunicarse entre las personas no es algo homogéneo ni fácil de definir. Cada persona tiene sus propias peculiaridades a la hora de relacionarse con su entorno mediante el uso del lenguaje. El poder crear un estándar de comunicación para el mayor número de individuos, tomando el castellano como idioma, será otro de los problemas que el proyecto desarrollado trate de solucionar. Mediante la obtención de datos recogidos en un log a través del uso de la aplicación desarrollada, se podrán analizar los usos que las personas hacen del lenguaje y crear un sistema de ayuda automática en situaciones de peligro.

1.3 Motivación

La dificultad para la simulación y el ensayo de situaciones de riesgo de la vida real es la motivación del proyecto y lo que se presenta a continuación. El proyecto desarrollado se centra en el campo de los entornos virtuales como herramienta para la obtención de datos aplicables a la vida real. Existe una amplia variedad de proyectos que trabajan sobre entornos virtuales para la recreación de situaciones reales. Esto se debe a los grandes resultados que se pueden obtener mediante el uso de dichos entornos.

Un ejemplo de este tipo de proyecto sería: "Creación de un entorno 3D para la simulación de tráfico urbano" (Pérez., 2009), el cual también se encuadrada dentro de otro proyecto del plan nacional. Este proyecto cuenta con un software capaz de captar imágenes mediante una cámara para poder detectar y distinguir las señales de tráfico.

El proyecto utiliza el entorno de pruebas Microsoft Robotics Developer Studio, que es un entorno basado en Windows para aplicaciones robóticas.

Mediante un entorno virtual pueden probar y recrear situaciones que podrían darse a lo largo de un día en cualquier carretera, valorar las consecuencias y sacar datos con los que prepararse para futuras necesidades.

El estudio y desarrollo de este tipo de proyectos está promovido por la fidelidad de los datos obtenidos sobre simulaciones virtuales de hechos reales. Un ejemplo serían las aplicaciones que se utilizan en la recreación de accidentes de tráfico, en las cuales, mediante la inserción de los factores conocidos recogidos en la escena; como pueden ser marcas de frenada, desperfectos ocasionados en el mobiliario urbano o la posición final del vehículo, son capaces de determinar factores que no son triviales a la hora de determinar la culpabilidad de los accidentados; como pueden ser la velocidad excesiva de alguno de ellos, una trayectoria invasiva del carril contrario, etc. Es en este campo donde la aplicación desarrollada toma su punto de partida.

El proyecto desarrollado toma como soporte virtual el modelado de la facultad de Informática de la universidad Complutense de Madrid. Gracias a este entorno y a los personajes proporcionados por la aplicación, el usuario podrá recorrer libremente el espacio de la facultad. Llegado a este punto podrá pedir que se le indique una ruta a seguir para llegar al lugar deseado. En el caso de tratarse de una situación de peligro provocada por un incendio el objetivo es proporcionar una ruta de escape sin riesgo para el o los afectados.

Una vez desarrollada la aplicación, los datos proporcionados por la misma podrán servir de base para revisión de los protocolos actuales de actuación en situaciones de peligro o la creación de nuevos. El poder mejorar dichos protocolos y minimizar el daño en situaciones provocadas por fuegos incontrolados son la motivación principal del proyecto presentado. Recrear este tipo de situaciones en la vida real, para aprender y mejorar los protocolos existentes es algo muy costoso, sin embargo, el proyecto desarrollado proporciona una herramienta donde simular estas situaciones las veces necesarias y adecuar los protocolos para hacerlos lo más eficientes posible. No olvidemos que el objetivo final es salvar el mayor número de vidas en una situación de peligro provocada por un incendio.

1.4 Objetivos

Los objetivos esperados del proyecto desarrollado son los siguientes:

- Desarrollar un entorno virtual de la facultad de Informática de la universidad Complutense de Madrid.
- Proporcionar un avatar con el que el usuario pueda interactuar con el entorno desarrollado.
- Simulación de fuegos en edificios cerrados
- Proporcionar los medios de comunicación entre el usuario (persona afectada en un fuego real) y el guía que le proporciona las indicaciones.
- Desarrollar un sistema cliente–servidor que gestione dicha comunicación entre los usuarios.
- Un entorno para la simulación de situaciones de riesgo provocadas por fuegos, con el que probar todas las posibilidades que se pudieran dar en la vida real.
- Recoger las conversaciones entre el usuario y el guía con el fin de estudiar la forma de comunicarse en esas situaciones.
- Proporcionar una herramienta con la que poder evaluar o modificar los protocolos de evacuación existentes.
- Minimizar el daño provocado por situaciones derivadas del fuego en espacios cerrados.

A lo largo del documento se irán presentando los objetivos anteriormente citados, las soluciones dadas para su resolución y la implementación llevada a cabo para conseguir cubrirlos.

También se presentarán los problemas surgidos durante el proceso de desarrollo y objetivos futuros interesantes. Estos objetivos aunque quedan fuera del proyecto desarrollado pueden aplicarse a futuras ampliaciones del mismo o a la creación de nuevas aplicaciones que los cubran. Un ejemplo sería la implementación de otro tipo de desastres como, inundaciones, terremotos, hundimientos de edificios, etc.

La dificultad de la creación de un estándar de comunicación radica en lo heterogéneo que es el uso particular del lenguaje por cada individuo. La riqueza gramatical del castellano incita esta cuestión, existen muchas formas de expresar una misma idea dependiendo de la persona que la exprese.

En el aspecto que ocupa al proyecto desarrollado, una misma indicación puede darse de diversas formas, por ejemplo, imaginemos que queremos indicar a alguien que salga por cierta puerta que encontrará si camina hacia delante, esto podría decirse como: “Diríjase de frente y salga por la segunda puerta”,”Avance y después de pasar la primera puerta salga por la siguiente puerta que se encuentre”, etc.

Encontrar la forma de comunicación más sencilla y eficiente a la hora de dar instrucciones será un objetivo importante del proyecto desarrollado.

1.5 Sobre el documento

Este documento pretende ser una ayuda para la creación y desarrollo de un entorno virtual enfocado a la obtención de datos para poder crear después sistemas automatizados de ayuda y desarrollar protocolos de evacuación efectivos.

Para ello se ha organizado el documento en cinco partes, en las que se comentarán el origen de los entornos virtuales, el origen de los problemas que trata este proyecto, las soluciones a dichos problemas y las conclusiones obtenidas.

En la introducción se presenta un resumen del proyecto y su encuadre dentro del proyecto nacional "NOVA". Se plantean los problemas que se pretenden resolver con el proyecto desarrollado, así como los objetivos que se esperan conseguir y la estructura del documento.

A continuación se mostrara el nacimiento y progreso de los entornos virtuales, así como ejemplos concretos de ellos, para posteriormente entrar en detalle de las herramientas con las cuales se pueden diseñar, eligiendo las adecuadas para diseñar este proyecto que se verá en la última parte de este capitulo.

Con las herramientas elegidas se crearan los prototipos que se muestran en el capítulo 3, exponiendo las características de cada uno, y sus ampliaciones respecto al anterior, para finalmente mostrar el resultado en la versión final.

La sección de la discusión presenta los objetivos alcanzados con el proyecto desarrollado, así como futuros pasos para desarrollar aplicaciones que cubran otras necesidades. También recoge aquellos aspectos relacionados con el proceso de desarrollo pero que quedan fuera del cuerpo principal del documento, como pueden ser, otros usos posibles de la aplicación.

En la sección de las conclusiones se resume el proyecto en líneas generales, se explican las soluciones aplicadas para la satisfacción de los objetivos, se hace un balance general y se enuncian diferentes aplicaciones y ampliaciones del proyecto.

Capítulo 2:
Estudio del estado del arte

2 Estudio del estado del arte

Si se habla de simulación de escenarios de evacuación, se está hablando también de realidad virtual. A partir de un entorno real, en este caso un edificio, se renderiza un modelo del mismo, un entorno virtual. El usuario de la aplicación tiene la sensación de encontrarse realmente dentro del edificio. Si además de permitir al usuario moverse libremente a través de este escenario, se utilizan periféricos que abstraigan sus sentidos, se estaría dando un salto de un entorno de realidad virtual semiinmersiva a otro de realidad virtual inmersiva. Si se integra el entorno virtual en el real, se trataría de un entorno de realidad aumentada. A lo largo de este capítulo se explicarán detalladamente todos estos conceptos, como han ido evolucionando en los últimos años y en qué estado se encuentran actualmente. Se verán ejemplos de otros entornos virtuales y finalmente se mostrarán diferentes motores gráficos y librerías con las cuales se puede crear y desarrollar un entorno virtual.

2.1 Realidad virtual

2.1.1 Definición

La realidad virtual es una representación de la realidad generada en tiempo real por un sistema informático en el que el usuario tiene la sensación de estar su interior, alejando los sentidos de la realidad. La interacción con el medio es implícita, es decir, el usuario no tiene que indicar a la máquina las acciones que quiere realizar, sino que es el sistema el que interpreta la forma natural de comportarse del usuario (Brunet, 2002).

Hay diferentes grados en los que se puede implementar la realidad virtual, dependiendo de la capacidad de interacción del usuario con el mundo virtual y los objetos del mismo. A medida que estos aumentan se avanza en el nivel de inmersión. La realidad virtual ideal sería aquella que desde una inmersión total nos permita una interacción sin límites con el mundo virtual, además de aportarnos como mínimo los mismos sentidos que tenemos en el mundo real (vista, oído, tacto, gusto, olfato). Sin embargo, la mayoría de los sistemas actuales se centran en únicamente 2 sentidos (vista y oído), debido a las dificultadas y costes de simular los demás.

Cascos, guantes y otros dispositivos capturan el movimiento y la rotación de las diferentes partes del cuerpo humano

En la realidad virtual inmersiva se consigue una inmersión total mediante periféricos (cascos de realidad virtual, gafas, posicionadores, HDM...), hasta el punto de desaparecer el mundo real. Presenta dos desventajas importantes: La inmersión mediante un casco impide al usuario ver su propio cuerpo, y por lo tanto, se hace difícil la manipulación de objetos en el entorno, ya que hay que utilizar una representación virtual de los brazos, y son muy difíciles de coordinar. Por otro lado, el hecho de no poder ver a las personas que hay alrededor dificulta la utilización del entorno para objetivos colaborativos.

La realidad virtual semiinmersiva permite observar el entorno virtual a través de una ventana, utilizando un monitor de alta resolución estándar. La interacción con el ambiente puede ocurrir por medios convencionales como teclados, ratones y trackball. Este tipo de realidad virtual es muy común en videojuegos y escenarios de simulación ya que no requiere ningún hardware especial, pero son poco útiles en aplicaciones donde es necesario sentir la escala de los objetos.

2.1.2 Historia:

Los inicios de la Realidad Virtual serán, dependiendo de las fuentes que se consulten, más o menos remotos, pero uno de los precedentes más claros es la industria del cine. Desde siempre la cinematografía ha intentado crear formatos de imagen y sonido que hiciesen creer al espectador que se encontraba formando parte de la escena. De este intento han surgido tecnologías como el Cinemascope o el Omnimax, Dolby Surround o recientemente el cine en 3D.

En 1929 se crea el primer Link Trainer (también conocido como Blue Box), que era un simulador de vuelo mecánico. Más de 500.000 norteamericanos fueron entrenados en simuladores basados en este modelo, más tarde se extendieron estos simuladores para estudiar las crecidas de ríos y presas.

En el año 1938, Charles Wheatstone, un inventor británico crea el primer estereoscopio, que consistía en una especie de gafas en las que se situaban 2 fotografías distintas en cada ojo, creando de este modo una imagen 3d en el interior del cerebro, o más bien la sensación de profundidad.

Dos décadas más tarde, en 1958, la empresa Philco Corporation crea un casco de realidad virtual que utiliza los movimientos de la cabeza del usuario para realizar los desplazamientos.

A partir de 1965 aparecieron una serie de innovaciones en el mundo de la realidad virtual de la mano de Ivan Sutherland. En este año describe el concepto de realidad virtual en un artículo titulado “The Ultimate Display” pero en el artículo no llega a utilizarse el término. Al año siguiente creo junto con otras personas un casco HMD. Los desplazamientos eran realizados con los movimientos de la cabeza. En 1967 funda con David Evans la empresa Evans & Sutherland y desarrollan el primer programa diseñado para crear mundo virtuales con imágenes 3d, datos almacenados y aceleradores. En 1968 la empresa Evans & Sutherland crea el primer casco estereoscópico.

A comienzos de los 70 se empezó a investigar cómo hacer más fácil el entendimiento hombre-computadora, para mejorar el rendimiento y obtener toda la potencia de estas máquinas, ya que mientras la capacidad y velocidad de los ordenadores aumentaba vertiginosamente, nuestra habilidad para comunicarnos con ellos permanecía limitada por interfaces inadecuados, Data Glove crea un guante de

datos que permite desplazarse por mundos virtuales y Frederick Brooks crea Grope II un sistema que permite visualizar moléculas complejas.

En 1979 se crea el primer simulador de vuelo basado únicamente en sistemas informáticos, dando origen a los primeros Entornos Virtuales. El simulador servía como material para una clase de aviación en el departamento de defensa de los Estados Unidos. Se comenzaron a apreciar las grandes ventajas de entrenar a pilotos de aviación en simuladores en lugar de emplear auténticos aviones: menores costes, reducción de tiempo y mejora del aprendizaje, además del consiguiente y obvio nivel de seguridad que impone la práctica virtual al no arriesgar la vida de los pilotos.

La comercialización de estos primeros Entornos Virtuales propició que se empezara a utilizar el término de Realidad Virtual, asociado normalmente a los videojuegos en los que el usuario interactúa con el entorno virtual mediante dispositivos de entrada/salida como guantes, gafas, cascos, etc.

Periféricos utilizados para videojuegos de realidad virtual

En 1980: La Compañía StereoGraphics crea las gafas de visión estéreo y en 1985 Scott Fisher, considerado uno de los "Padres Fundadores" de la realidad creó el VISIOCASCO más avanzado en la Nasa Ames Center. Por todas partes empiezan a surgir equipos de desarrollo trabajando en lo que era la tecnología de la realidad virtual, y se empiezan a ver los primeros resultados comerciales. En el mismo año Thomas Zimmerman patenta un Electro guante que inventó mientras investigaba sobre cómo controlar con la mano un instrumento musical virtual.

Durante los años 80 se siguió innovando en el campo de la realidad virtual, la compañía Inglesa Dimensión Internacional desarrolla un software de construcción de mundos tridimensionales y se inventan dispositivos para la generación de sonido tridimensional, ATARI saca al mercado la primera máquina de galería de videojuegos con tecnología 3D. Autodesk presenta su primer sistema de realidad virtual para P.C.

Ya en 1994 aparece la primera versión del VRML (Virtual Reality Modeling Language) para representación de escenas y objetos 3D en la Web y a partir de aquí entramos de lleno a la carrera comercial. Los sistemas de realidad virtual comienzan a popularizarse y muchos productos empiezan a invadir el mercado.

2.1.3 Tipos de dispositivos: Hacia la inmersión total

Para dar al usuario la sensación de que forma parte del entorno hace falta abstraer sus sentidos, y para ello se usan dispositivos que permitan manejar los objetos virtuales como se manejarían los objetos en el mundo real, de forma intuitiva y sin aprendizaje previo. Como indicábamos anteriormente, el desarrollo de estos dispositivos se ve frenado por la dificultad y el coste de ir más allá de simular los sentidos de la vista y el oído. Sin embargo, hay multitud de sistemas que tratan de acercarnos cada día más al ideal de la inmersión total.

Podemos separar los dispositivos en dos clases, sensores, que tienen como objetivo captar las acciones del participante y transmitir esa información al sistema, y los dispositivos de salida o efectores, que generan los estímulos necesarios para provocar el efecto inmersivo en los sentidos del usuario (Silva, 2007).

Los audífonos son el equipo básico empleado para escuchar los sonidos propios de un ambiente virtual. Con los denominados audífonos convencionales, los de uso más corriente, se escucha el sonido simulado de los objetos sin identificar auditivamente el punto de ubicación de los mismos. Utilizando audífonos especiales, como el convolvotrón, además de simular el sonido propio de los objetos, se puede simular la ubicación de los mismos dentro del ambiente virtual (Jose R. Hilera, s.f.).

Encontramos los HMD's o Head Mounted Display (Display Montado en la Cabeza). Estos abarcan desde cascos completos a ligeras gafas de solo una pantalla.

Sony patentó la idea de poder transmitir olores, sonidos e incluso sentimientos realizando una transmisión directamente al cerebro mediante ultrasonidos, y científicos del Shinoda lab (Horsnell, 2005) de la Universidad de Tokio han desarrollado hologramas 3D. Algunos de los proyectos más ambiciosos que se han realizado últimamente en este campo harán posible la locomoción en entornos de realidad virtual.

Cinta andadora que permite el movimiento en cualquier dirección

Las caminadoras habituales permiten solamente andar hacia delante o hacia detrás, sin embargo, el proyecto europeo Cyberwalk ha desarrollado cybercarpet, que permite andar en cualquier dirección. Este avance, que mostramos en la figura superior, permitirá aplicar la realidad virtual para terapias, entrenamiento o eventualmente para videojuegos. (Tecayehuatl, 2009).

Se ha desarrollado sistemas para simular la locomoción como la esfera virtual

Otra posibilidad sería Virtusphere, que permite al usuario una mayor libertad de acciones tales como correr, saltar, o incluso arrastrarse. El mayor handicap sería su precio (35.000$) y su escasa aplicación, que se limita al terreno militar y el mundo de los videojuegos.

2.1.4 Usos y aplicaciones

Durante años, los diferentes diseños de Realidad virtual han sido implementados siguiendo un formato estandarizado para representar imágenes o mundos 3D, VRML: Virtual Reality Modelling Language (Lenguaje Modelado Realidad Virtual) (RV, s.f.).

Este formato fue diseñado para ser utilizado en la web. La primera versión del mismo fue especificada en 1994, y en 1997 una nueva versión llamada VRML2 o VRML97 fue presentada. Actualmente está en desuso y ha sido sustituido por el X3D.

La aplicación de la realidad virtual no solo afecta al mundo de la ingeniería, sino que también se extiende sobre una gran cantidad de áreas como la medicina, la arquitectura, la educación y la ingeniería entre otros. Los usos actuales más frecuentes de la realidad virtual son los siguientes:

En el ámbito de la arquitectura, el interiorismo y la planificación urbanística, la principal ventaja que se ofrece es el hecho de poder experimentar con diversas distribuciones de los distintos elementos del edificio o el mobiliario.

También pueden usarse para medicina educativa, por ejemplo para la planificación de operaciones quirúrgicas y realización de diagnósticos no invasivos.

CAD (diseños asistidos por ordenador). Permite ver e interactuar con objetos antes de ser creados, con el consiguiente ahorro de costes.

También puede aplicarse al entrenamiento de pilotos, astronautas, soldados, a la creación de entornos virtuales en museos, tiendas, aulas, ferias juegos y por supuesto al cine 3D y todo tipo de entretenimiento.

Realidad virtual aplicada a la simulación militar, en la imagen, paracaidismo

2.1.5 CAVE, experimentando la Realidad Virtual

CAVE es el acrónimo de "Cave Automatic Virtual Environment" y es sistema virtual de proyección que representa un entorno de realidad virtual inmersiva. El usuario se sitúa en el interior de un cubo, y son proyectadas en cada una de las paredes imágenes desde el exterior (RV, s.f.). Existen CAVEs en los que se proyecta las imágenes en 3, 4, 5 o 6 caras del cubo.

Generalmente el usuario lleva una gafas estereoscópicas que sincronizadas con las proyecciones le permiten ver en 3D las imágenes, e incluso contemplar objetos flotando dentro del cubo que puede observar desde todos los ángulos. La interacción y la navegación en CAVE son posibles a través de un ratón tridimensional. Como podemos observar en la imagen inferior.

Cave: realidad virtual inmersiva

El primer CAVE fue creado por el equipo EVL (Electronic Visualization Laboratory) de la Universidad de Illinois (Chicago-USA), y fue presentado por primera vez en 1992 en la conferencia anual SIGGRAPH.

2.2 Realidad Virtual integrada en Entornos Reales: Realidad Aumentada:

2.2.1 Descripción del concepto de realidad aumentada

Realidad Aumentada (RA) es un término acuñado por Tom Caudell en el año 1990 y describe la aumentación de la realidad física mediante el uso de técnicas que la mezclan con contenido digital (virtual). Tenemos pues que la Realidad Aumentada es un Entorno que incluye elementos de los dos mundos (virtual y real), es interactivo en tiempo real y se muestra en tres dimensiones; podemos decir que se construyen nuevos mundos mixtos coherentes con ambos sistemas (Bonnin, s.f.).

Actualmente hay dos definiciones mayoritariamente aceptadas, la de P.Milgram & F.Kishino y la de R.Azuma.

La definición creada por Paul Milgram y Fumio Kishino en 1994 llamada Milgram-Virtuality Continuum dice que entre un entorno real y un entorno virtual puro esta la llamada realidad mixta y esta se subdivide en 2, la realidad aumentada (más cercana a la realidad) y la virtualidad aumentada (más próxima a la virtualidad pura).

Este gráfico define mejor el concepto explicado:

Y la definición aportada por Ronald Azuma en 1997, que acota a la realidad aumentada a la que cumple estos tres requisitos: combinación de elementos virtuales y reales, interactividad en tiempo real e información almacenada en 3D.

El auge actual es debido principalmente a dos razones: la primera es el gran desarrollo de los interfaces gráficos y la segunda son la disponibilidad de dispositivos portátiles (PCs, PDAs, terminales móviles). Estos dos avances han conseguido eliminar los molestos dispositivos montados en la cabeza (cascos con cámara, gafas) y los trajes especiales que tanto limitaban, por su parafernalia, el uso de los mismos.

El efecto de la realidad aumentada se consigue a través de un dispositivo que cuente como mínimo con la combinación de una cámara y una pantalla. Al visualizar la realidad a través de la pantalla (de un móvil, o unas gafas) observamos la misma realidad en tiempo real, pero el dispositivo se encarga de añadir información adicional como por ejemplo:

Turismo: Al observar un paisaje el dispositivo nos puede informar de lugares que podemos visitar y a qué distancia estamos de los mismos.

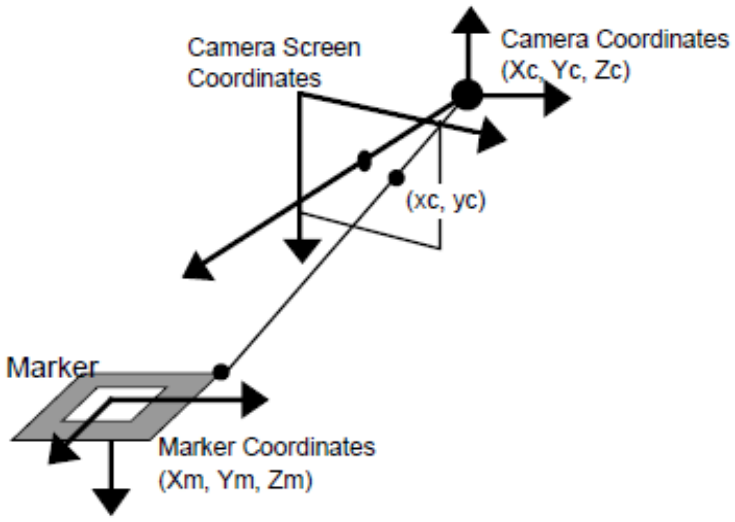
Objetos Virtuales: Una de las formas de implementarlo sería mediante un trozo de papel en el que hay dibujado una figura geométrica simple, el software puede recrear un objeto virtual y este ser añadido a la realidad para ser observado. Existen en la actualidad aplicaciones útiles en marcha que van desde los juegos, a probadores de relojes.

Educación: Por ejemplo al visitar un museo podemos visualizar un cuadro o una escultura a través del dispositivo que estemos utilizando y este nos ofrecerá información en pantalla acerca del autor, fecha, estilo, etc.

Medicina: A un cirujano le puede aportar información útil en tiempo real, como por ejemplo el número de latidos, temperatura, bordes de un tumor, etc.

Aplicación de Realidad Aumentada para Iphone: ***Star Wars Arcade: Falcon Gunner***

A finales de los años noventa fueron desarrolladas las primeras librerías de realidad aumentada, ARtoolkit (Augmented REality Toolkit), de Kato y Billinghurst (1999). Este sistema analiza la imagen grabada por la cámara de vídeo en busca de un patrón gráfico (un marco cuadrado negro, con diferente diseño en su interior para calcular la orientación espacial del objeto), partiendo de la posición original que grabó la imagen. Cuando este patrón es detectado, son analizadas las coordenadas de sus cuatro esquinas y gracias a los valores de cada uno de esos cuatro puntos espaciales colocados sobre un plano obtiene la posición y orientación en el espacio cartesiano de la cámara que tomó la imagen relativa a la marca.



Relaciones entre la posición del marcador RA y la cámara

2.2.2 Hardware

Para poder utilizar la realidad aumentada en necesario un dispositivo que cuente con las siguientes características:

Sistema de visualización: Por lo general una pantalla que está situada en un dispositivo de mano (como por ejemplo un smartphone), o montada en un casco o gafas.

Dispositivo de entrada: Como mínimo ha de contar con una cámara, aunque resulta útil disponer de algún dispositivo que nos sitúe en el espacio (por ejemplo un GPS). Otros dispositivos de utilidad son los compases de estado sólido, los giroscopios y los acelerómetros.

CPU: Es necesario un potente procesador para poder manejar y modificar el video en tiempo real.

El funcionamiento con estos dispositivos consiste en situar el dispositivo entre la realidad y nuestros ojos y de este modo ver la realidad aumentada a través de la pantalla del mismo.

La mayoría de estos requisitos los cumplen los actuales teléfonos móviles inteligentes o también llamados smartphones, por lo que son una plataforma muy atractiva para los desarrolladores de aplicaciones, ya que nos permiten utilizar esta tecnología sin tener que comprar un nuevo dispositivo.

2.3 Entorno virtual

2.3.1 Concepto de entorno virtual

Un entorno virtual es una representación gráfica de un modelo del mundo real que se consigue mediante un proceso de renderizado, donde se genera una imagen de 2D a partir de una en 3D. Esta imagen puede mostrarse en 2D o desdoblarse en dos imágenes, una para cada ojo y mostrar el resultado en 3D haciendo uso de técnicas de fotografía estereoscópica.

Tienen su origen en la simulación militar y en concreto en los simuladores de vuelo, y tuvieron un impulso muy importante gracias al rápido desarrollo de la tecnología gráfica 3D, y de hardware dedicado al procesamiento de esta información. Pronto estuvieron disponibles para el consumidor de videojuegos algunos productos de una calidad visual cercana a la de los simuladores militares (Miguel Lozano, s.f.).

Poco tiempo después aparecieron los entornos virtuales inteligentes, basados en los entornos virtuales convencionales, pero incluyendo además técnicas de inteligencia artificial. El sistema de transporte de Londres, capaz de ubicar incidencias en tiempo real en el mapa de metro londinense, ha recurrido a estas tecnologías para ofrecer nuevos servicios en tiempo real a sus clientes.

También se han utilizado entornos virtuales para representar ciudades en 3D, con objetivos turísticos, como es el caso 3D que detallaremos más adelante, o Urbamedia (Arturo F. Montagu, s.f.) que utiliza un modelo en 3D de la ciudad de Buenos Aires para medir diferentes tipos de impactos (ambientales, visuales, sonoros y nuevas normas urbanísticas).

2.3.2 Tendencias actuales de los entornos virtuales

2.3.2.1 Second Life:

Second life es el entorno virtual 3D más activo del mundo, con cerca de 16 millones de usuarios y cerca de 300.000 visitas diarias, donde España ocupa la novena posición en el ranking de usuarios activos por países (Elvira San Millán Fernández, 2008). En palabras del desarrollador de la idea Philip Roselade, de Linden Lab, se trata de "una revolucionaria nueva forma de experiencias compartidas, donde los individuos se reúnen en una tierra inhabitada en 3D para construir el mundo alrededor de ellos". Proyecta al usuario en una plataforma tecnológica tridimensional permitiendo la reconstrucción de la propia personalidad.

2.3.2.2 Aplicaciones en educación de los entornos virtuales

Las utilidades de los entornos virtuales se está aplicando en las áreas de educación y pedagogía, como por ejemplo en la Harvard Law School, donde se basa en el entorno Second Life para permitir a los alumnos interactuar directamente y participar en la creación de un argumento que han de defender ante un tribunal, con otro grupo de estudiantes que actúan como jueces.

De esta forma se abre la posibilidad de impartir lecciones didácticas donde acudan estudiantes de todo el mundo y puedan realizar prácticas de anatomía, simulación de urgencias en hospitales o entrenamiento de bomberos, donde los estudiantes ponen a prueba su capacidad de reacción, adquieren nuevas habilidades y toman decisiones.

El entorno es utilizado por bomberos, policías, personal sanitario, etc. para saber cómo tienen que comportarse y qué hacer en caso de vivir las distintas situaciones que se plantean. Además, no solo se aprende cómo actuar en situaciones extremas, sino que en el entorno virtual de Play2train hay disponibles diferentes herramientas y sistemas de aprendizaje de habilidades concretas (aprender a evacuar, tratamientos críticos, admisión, instrumental médico...).

Aprendizaje de idiomas: Second Life ofrece un poderoso escenario basado en contextos para el aprendizaje de idiomas. En LanguageLab, por ejemplo, podemos acceder directamente a los escenarios virtuales 3D de un aeropuerto, un hotel o un banco, y enfrentarnos a las típicas situaciones con las que nos encontraríamos en estos lugares.

2.3.2.3 Turismo virtual: Medisolae 3D

Grecia lidera un proyecto euro mediterráneo de turismo virtual. La empresa Epsilon está desarrollando una experiencia turística virtual del Mediterráneo (Epsilon, s.f.). El objetivo es recrear en virtualmente cien islas mediterráneas para que el turista pueda preparar de manera lo más visual posible las rutas de su próximo destino turístico.

El proyecto, bautizado Medisolae 3D, utilizará el sistema de armonización de datos geográficos europeos, Inspire, para crear un entorno de navegación simulada. La información se vinculará con los conocidos sistemas Google Earth, Virtual Earth y ArcGlobe y, también, con centrales de compras de reservas hoteleras y servicios vinculados al turismo. El resultado, una experiencia completa que ayudará a los turistas a preparar sus viajes sobre el terreno. Cofinanciado por la Unión Europea e impulsado por un consorcio de 14 empresas provenientes de seis países, entre las que se encuentra la española Fundación Illes Balears (IBIT), Medisolae 3D permitirá reproducir entornos naturales, arqueológicos, y urbanos de las principales islas del Mediterráneo, como Santorini, Thasos, Salamina, Kefalonia, Mykonos, Chipre, Groix, Aix, Capri, Sicilia, Mallorca, Formentera, Menorca, Ibiza y Cabrera, entre otras.

A nivel tecnológico, Medisolae 3D es un vasto proyecto que implica la modelización en 3D de edificios y zonas históricas que serán integrados y texturizados gracias a fotos para ser visualizados en la navegación virtual.

El proyecto Medisolae permite visitar islas mediterráneas de forma virtual

2.3.3 Visión estereoscópica de un entorno virtual: Nintendo 3DS

La estereoscopía es cualquier técnica capaz de recoger información tridimensional o de crear sensación de profundidad en una imagen. La ilusión de profundidad se consigue presentando dos imágenes bidimensionales diferentes con una pequeña desviación, una para cada ojo, imitando la forma natural de recoger la realidad.

Recientemente ha salido al mercado la videoconsola Nintendo 3DS que permite al usuario percibir imágenes 3D sin la necesidad de usar unas gafas accesorias, utilizando una tecnología desarrollada por Sharp. Se crean unas líneas verticales que dirigen la atención de los ojos creando la sensación de tridimensionalidad. Este sistema solamente funciona cuando la imagen se mira de frente, por lo que puede ser aplicada también a smartphones o portátiles, pero no tendría sentido para los televisores convencionales (Nintendo, 2011).

Nintendo 3DS permite ver imágenes en 3D emitiendo simultáneamente una imagen para cada ojo

2.4 Otros entornos virtuales:

2.4.1 Creación de un entorno 3D para la simulación de tráfico urbano

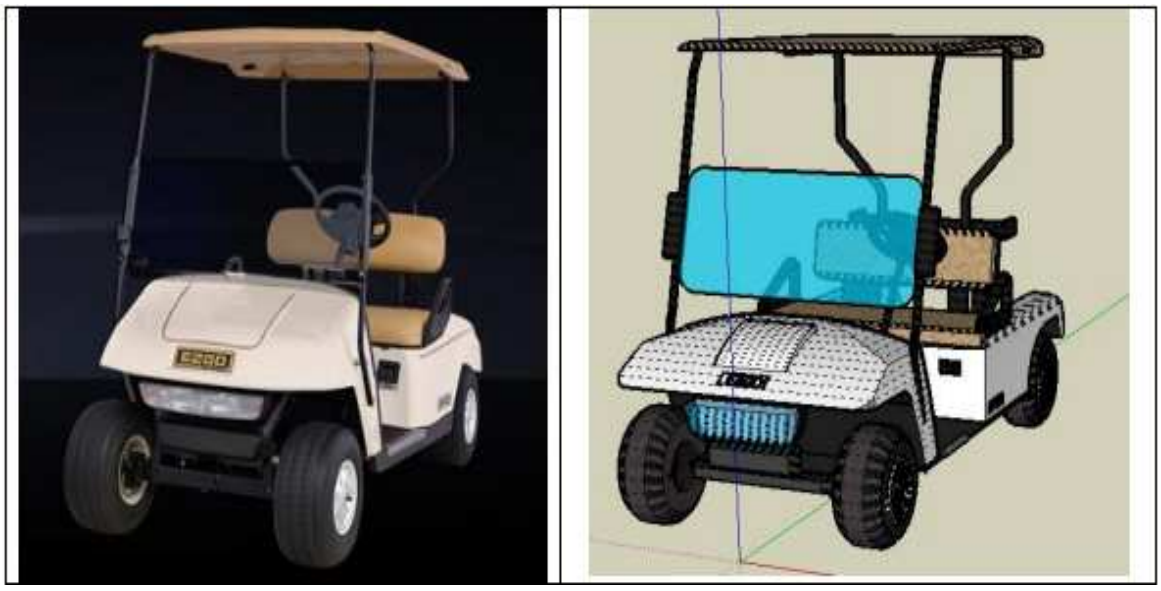
Es un proyecto de fin de carrera para la universidad Carlos III de Madrid realizado por Víctor Romero Pérez. (Pérez, 2009).

Los departamentos de Informática y de Industriales de la escuela Politécnica de la Universidad Carlos III de Madrid desarrollaban un proyecto de investigación centrado en técnicas de fusión sensorial, donde la plataforma de experimentación consiste en un carrito de golf equipado con una serie de sensores, cada uno de los cuales estará encargado de recopilar un determinado tipo de información. Esta a su vez enviara a una unidad central, la cual se encargará de procesar toda esta información y a partir de ella desarrollar sistemas de asistencia a la conducción.

Dentro de este proyecto se tiene ya desarrollado un software que mediante imágenes captadas por una cámara, es capaz de detectar y distinguir las distintas señales verticales de tráfico que se pueden encontrar en la vía pública.

Antes de implementarlo y probarlo en el mundo real, se pretende realizar una serie de pruebas en un entorno simulado. El entorno de pruebas que utiliza es Microsoft Robotics Developer Studio, que es un entorno basado en Windows para el desarrollo de aplicaciones de robótica.

En esta parte del proyecto es en la que se integra este proyecto de fin de carrera, que consiste en modelar, un carrito de golf y otras entidades en tres dimensiones (3D) mediante un software con el que se pueda importar el resultado e integrarlo en Robotics Developer Studio, para así poder realizar pruebas sobre el correcto funcionamiento del software ya desarrollado.



Versión 3D del carrito de golf usado para recopilar los datos.

2.4.2 Proyecto <e-Adventure 3D>: Entorno de Autoría Para la Creación de Aventuras 3D Educativas en Entornos Virtuales de Enseñanza

Es otro proyecto de fin de carrera en este caso de la Facultad Informática de la Universidad Complutense de Madrid (Torrente Vigil, Cañizal Alzola, & del Blanco Aguado, 2007/2008).

El proyecto trata de aportar una plataforma intuitiva pero completa y potente para el desarrollo de aventuras 3D con fines educativos con un coste razonable. Con tal propósito la plataforma consta de dos aplicaciones, la herramienta de edición orientada a autor y el motor de juego. El diseñador del juego (probablemente un profesor o instructor) toma los recursos producidos por los artistas (modelos 3D, música, imágenes, etc.) y desarrolla el juego con el editor. Por su lado el motor de juego es el encargado de ejecutar dichos juegos. Los educadores no necesitan tener ningún conocimiento técnico sobre el desarrollo de videojuegos, simplemente han de centrarse en los aspectos educativos. Por esta razón la plataforma <e-Adventure3D> incluye características para la evaluación automática del alumno (conocido como "assessment"), así como herramientas para ajustar el comportamiento de los juegos dependiendo de distintos factores, en lo que se conoce como "adaptation" (no todos los estudiantes tienen la misma forma de aprender).

Ejemplo aplicación creada con e-Adventure 3D

2.5 Herramientas

2.5.1 Motor grafico

La funcionalidad básica de un motor es proveer al videojuego de un motor de renderizado para los gráficos 2D y 3D, motor físico o detector de colisiones, sonidos, scripting, animación, inteligencia artificial, redes, streaming, administración de memoria y un escenario gráfico. El proceso de desarrollo de una aplicaciones puede variar notablemente por reusar o adaptar un mismo motor de aplicaciones 2D y ·3D.

Muchos motores están destinados sobre todo a la creación de videojuegos, que son aplicaciones que tienen que manejar una gran cantidad de recursos.

Algunos ejemplos de motores gráficos son SourceEngine, Rage o Unreal Engine3.

SourceEngine (SourceEngine). Usado en Half-Life 2, TeamFortress 2 y algunas joyas de otras empresas como ZenoClash. Aunque ha estado entre nosotros desde hace mucho tiempo (2004) y, es probablemente el más viejo de esta generación, aún se mantiene bastante funcional, como puede evidenciar la inminente salida de Left 4 Dead 2. El Source Engine todavía se destaca por una buena iluminación dinámica, compensación de lag en conexiones, un motor de físicas eficiente en conexiones de Internet y, sobre todas las cosas, un código fuente con simple acceso para los mods.

Rage (Rage)- Uno de los motores más importantes de la última década. GTA IV fue un gran juego en sí, pero su tecnología no se quedaba atrás. Curiosamente, el último GTA fue el primero en usar el motor, ya que no existía tecnología capaz de correr lo que tenían pensado hacer para su próximo juego, Red Dead Redemption. (Así fue como nació RAGE). Este motor es polifuncional en muchos sentidos, ya que puede manejar grandes mundos abiertos con muchos interiores, inteligencia artificial compleja, efectos de clima en tiempo real y distintos tipos de interfaces. Pero una de las mejores cosas es su integración con el espectacular motor de físicas, Euphoria, de la compañía NaturalMotion.

UnrealEngine3 (UnrealEngine3). Tal vez no sea el mejor, pero por alguna razón es el más licenciado por las compañías más poderosas en cuanto a desarrollo. EpicGames, los responsables del motor, han hecho juegos como Gears of War, pero muchas otras compañías lo han licenciado y modificado a puntos irreconocibles. Los mejores ejemplos de este caso son MassEffect y Bioshock. Más allá de cualquier problema, el Unreal Engine es el que mejor funciona en cualquier plataforma y encima ofrece excelentes gráficos con el motor de físicas Havok y una excelente fidelidad a la hora de jugar en línea. Fácilmente el más completo y sólido de todos.

Ogre3D (Junker, 2006)- Es un motor de renderizado de gráficos de código abierto que está escrito y mantenido por un pequeño equipo, y contribuye a él su creciente comunidad. Es uno de los mejores motores libres actualmente.

Ogre es un motor gráfico, y SOLO un motor gráfico. Sin embargo, puede emparejarse fácilmente con otras librerías para crear un motor con todas las necesidades cubiertas (jacmoe, 2010). Algunas librerías que se podrían necesitar para crear una aplicación 3D son:

- Sonido
- Redes
- Entrada
- Colisiones

Ogre no incluye estas librerías nativamente, sin embargo expone una interfaz que hace fácil trabajar en Ogre en las aplicaciones existentes. Hay varias razones por las que Sinbad (el líder del proyecto) escogió este camino. Y aquí su explicación:

"OGRE es un componente en un largo sistema de desarrollo. OGRE no es, y nunca fue, pensado como una plataforma para el desarrollo de juegos, esto es una herramienta para un propósito específico"

A continuación se comentaran alguna de las características más importante que tiene Ogre3D (Ogre).

2.5.1.1 Características para la Productividad

Uso de interfaces diseñadas para minimizar el esfuerzo de renderizar escenas 3D, y ser independiente de la implementación 3D. Ofrece un Framework de Ejemplo extensible que hace que tu aplicación funcione rápida y fácilmente. Requisitos comunes como el manejo del estado de renderizado, culling jerárquico, el manejo de las transparencias es automático para salvar, tu valioso tiempo. Ofrece un diseño limpio y bien documentado para todas las clases del motor. Además está orientado a objetos que permite extender la funcionalidad del motor a través de plugins y subclases con muy poco esfuerzo.

2.5.1.2 Plataforma y Soporte del API 3D

Tiene soporte para DirectX (Soporta DirectX9 y 10) y OpenGL [DirectX 7 no es soportado desde Ogre 1.2.0]. Además es multiplataforma, funciona en Windows (todas las versiones), Linux y Mac OSX. Se puede compilar en Visual C++ 2003, 2005, 2008 y 2010. También en gcc 4+ en Linux y Max OS X y para el iPhone.

2.5.1.3 Soporte para Material/ Sombreador

Es un potente lenguaje de declaración de materiales que te permite mantener varios conjuntos de materiales fuera de tu código. Tiene un soporte para el sombreado de vértices y programas de fragmentos (sombreadores), ambos a bajo nivel y escritos en ensamblador, y programas escritos con: Cg, DirectX 9 HLSL, u OpenGL GLSL. Soporta un completo rango de operaciones, como son multitexturas y mezclado multipasada, generación y modificación de coordenadas de textura, operaciones de color y alpha independientes para hardware no programable o para materiales de bajo costo.

Permite usar técnicas múltiples para el material lo que significa que puedes diseñar el material con efectos alternativos para un amplio rango de tarjetas y OGRE automáticamente usará el método mejor soportado. Los materiales pueden reducir el número de objetos necesarios, gracias a su soporte LOD de material.

Permite carga texturas desde PNG, JPEG, TGA, BMP, PVRTC o DDS, incluyendo formatos inusuales como texturas 1D, texturas volumétricas, cubemaps, HDR (alto rango dinámico) y texturas comprimidas (DXT/S3TC). También soporta la carga de texturas dinámicas en todos los formatos soportados, para reproducción eficiente de películas y otro contenido en tiempo real sobre la textura.

2.5.1.4 Mallas

Formatos de datos de malla flexibles, separación de los conceptos de vértices de búfer, índices de búfer, declaraciones de vértices y buffer mappings. Se pueden exportar desde muchas herramientas de modelado como Milkshape3D, 3D Studio Max, Maya, Blender y Wings3D

Las mallas tienen animación Esquelética, incluyendo el mezclado de múltiples animaciones y uso de huesos de tamaño variable.

Por otra parte también tiene parches bicuadráticos bezier para superficies curvas y Progresivas (LOD)

Además se pueden crear primitivas al estilo de OpenGL

2.5.1.5 Características de Escena

Ofrece un manejo de la escena de forma altamente personalizable y flexible, no con un sólo tipo de escena. Usa clases predefinidas para la organización de la escena. Existe un manejo genérico jerárquico de la SceneManager para límites volumétricos

Dispone de varios plugins para la creación de escenas. Uno de ellos está basado en BSP que permite rápidos renderizados en interiores, cargando niveles de Quake3 incluyendo el soporte para pasar scripts de sombreador, el plugins Octree que te permite generar escenas genéricas basadas en octree y el plugins de Terreno que te permite renderizar terrenos geo-mipmapped.

Los gráficos de escena son jerárquicos; nodos que le permiten a los objetos ser enlazados unos con otros y seguir sus movimientos, estructuras articuladas, etc.

Tiene un ajuste de ruta spline para objetos de escena, incluyendo entidades y cámaras, las cuales pueden ser entonces animadas fácilmente.

2.5.1.6 Efectos Especiales

Ogre cuenta con unos sistemas de partículas, incluyendo emisores extensibles y afectores (personalizables a través de plugins). Los sistemas pueden ser definidos en scripts de texto para un fácil tuneado. Uso automático de piscina de partículas para máximo rendimiento

Soporta cajas celestes, planos de cielo y dominios celestes, de muy fácil uso.

Se pueden usar objetos transparentes automáticamente manejados (orden de renderizado y profundidad de búfer ajustados todos para ti). Dispone también de un control flexible de niebla

Además tiene un sistema de superposición que te permite construir HUDs y menús usando objetos 2D y 3D.

Permite la carga de modelos compuestos de muchas mallas gracias al Software para dot3 bump mapping. También dispone de Cube mapping

Tras el Post-Procesado se pueden agregar efectos

2.5.1.7 Características Varias

Tiene una infraestructura de recursos comunes para el manejo de la memoria y la carga desde archivos (ZIP, PK3), una arquitectura flexible de plugins que permite al motor ser extendido sin recompilarse y "Controllers" que te permiten organizar valores derivados entre objetos ej: cambiando el color de una nave basando en el valor del escudo.

Permite depurar el manejo de la memoria para identificar las fugas de memoria

Cuenta con una herramienta llamada XMLConverter para convertir eficientemente binario a y desde XML para intercambio o edición.

Se puede inicializar Ogre con capacidades propias de tu GPU para probar y personalizar tu aplicación OGRE y como debería aparecer en hardware antiguo.

Dispone de un framework que puede integrarse con otras librerías tales como física y colisión mediante ReferenceAppLayer. Esto te da la flexibilidad de usar las librerías que quieras, en vez de estar preso de una. Mucha gente está usando ODE, Tokamok (vía Ogretok), Newton, ODE y OPCODE para detección de física y colisiones, pero puedes usar la librería que quieras.

Ogre estaba pensado en un principio para ser usado con C++, pero en la actualidad existen varios wrappers que posibilitan usar otros lenguajes de programación en vez de C++. Hay actualmente wrappers para Python (Python-Ogre), Java (Ogre4j) y para lenguajes .NET - C# y VB.NET - (MOGRE).

De entre ellos, una buena opción es usar Python (Python), debido a su sencillez para desarrollar código, siendo así muchísimo más productivo. Python es un lenguaje de programación fácil de aprender y potente. Tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos simple pero eficaz. La elegante sintaxis de Python, su gestión de tipos dinámica y su naturaleza interpretada hacen de él el lenguaje ideal para guiones (scripts) y desarrollo rápido de aplicaciones, en muchas áreas y en la mayoría de las plataformas.

Al ser un lenguaje dinámico, el rendimiento debería ser inferior al obtenido si se usase un lenguaje estático como C++. Pero esto es inapreciable debido a que Python se usa solo en una capa muy externa, que lo que hace es llamar a funciones nativas de C++, aprovechando así todo el potencial de C++, pero usando un lenguaje tan sencillo como es Python.

2.5.2 Motor de física

Como se comentó anteriormente Ogre es un motor gráfico, y SOLO un motor gráfico, por lo tanto se necesita usar otras librerías para complementarlo.

Una parte muy importante es la física. Gracias a ella se puede simular situaciones reales como son la gravedad del entorno, el choque de personajes con el entorno, y con otros personajes. Ogre dispone de varias librerías para implementar la física en él.

Ogre Bullet: wrapper de la mastodóntica biblioteca de físicas Bullet para Ogre3D. Es multiplataforma (disponible en decenas de sistemas) y ofrece todas las funcionalidades que uno podría desear con una eficiencia envidiable.

Ogre ODE: el correspondiente wrapper de Open Dynamics Engine para Ogre3D. Básicamente presenta las mismas ventajas e inconvenientes que la anterior.

MOC: Minimal Ogre Collision es una biblioteca tremendamente sencilla que proporciona al usuario lo mínimo para detectar colisiones. Se compone únicamente de dos ficheros sólo hay que añadir al proyecto. El problema es que obliga al uso del gestor de terrero ETM y no incluye callbacks.

Beastie: sencilla biblioteca compuesta únicamente por un fichero de cabecera enfocada a la detección de colisiones entre formas básicas y raycasting (emisión de rayos sobre la escena). Su interfaz no me acababa de convencer y tampoco ofrece callbacks ni cuerpos compuestos de varias formas.

2.5.3 Sonido

OgreAL es un wrapper para OpenAL que permite a los usuarios integrar a la perfección la música y el audio en las y aplicaciones de Ogre.

OpenAL (OpenAl)es una Api de audio multiplataforma desarrollada por Creative Labs para el renderizado eficiente de audio posicional y multicanal en tres dimensiones Está ideada para su uso en videojuegos y el estilo y convenciones del código es similar al de OpenGL.

OgreAL actualmente se puede manejar tanto los formatos WAV y OGG Vorbis de audio y es totalmente compatible con OpenAL 1.1 y permite al usuario acceder a cualquier funcionalidad que ofrece el Núcleo OpenAL. Esto incluye:

- Música de fondo
- Posicionamiento 3D .Esto permite al usuario personalizar al máximo la atenuación de la distancia.
- Multi-canal de audio
- Sonido direccional Este utiliza un sistema de doble cono que permite al usuario un control detallado sobre el sonido final.
- El efecto Doppler El algoritmo de Doppler tiene en cuenta la distancia, velocidad y dirección para calcular el resultado final.

Limitaciones

Hay un número limitado de fuentes permitidas en el hardware de audio. Este límite varía de tarjeta a tarjeta y en la actualidad OgreAL limita el número de sonidos.

Para encontrar este límite en el hardware actual puede llamar SoundManager::maxSources(). Hay un plan para solucionar este problema en un futuro próximo y así permitir infinitos sonidos.

La única pega que tiene esta librería es esta limitación, pero como en el entorno virtual el número de fuentes no será muy elevado, hace que la librería sea perfecta para el proyecto

2.5.4 Red

Ogre dispone de una librería para el trabajo en red muy potente llamada Raknet.Raknet es un Engine en c++ orientado al trabajo en red, está diseñado para tener un alto rendimiento, fácil integración y es una solución principalmente orientada al desarrollo de videojuegos. Raknet ha liberado una versión Free for Indie Developers bajo unas ciertas condiciones.

- Ganancias del juego inferiores a $250.000 USD.
- Debe mostrar el logo de Raknet en las pantallas iniciales
- Jenkins Software conserva la propiedad intelectual del código fuente de Raknet y sus distribuciones de desarrollo en línea (no es Open Source). Sin embargo los productos derivados son propiedad del desarrollador Indie.

Estas limitaciones hacen que para algunas aplicaciones haya que buscar una alternativa, como es el uso de los socket, los cuales se pueden implementarse bajo múltiples lenguajes (Python los acepta) de una manera segura y rápida.

Un socket queda definido por la dirección IP de la máquina, el puerto en el que escucha, y el protocolo que utiliza. Los tipos y funciones necesarios para trabajar con sockets se encuentran en Python en el módulo socket.

Los sockets se clasifican en sockets de flujo (socket.SOCK_STREAM) o sockets de datagramas (socket.SOCK_DGRAM) dependiendo de si el servicio utiliza TCP, que es orientado a conexión y fiable, o UDP, respectivamente. Los sockets también se pueden clasificar según la familia. Tenemos sockets UNIX (socket.AF_UNIX) que se crearon antes de la concepción de las redes y se basan en ficheros, sockets socket.AF_INET, socketssocket.AF_INET6 para IPv6, etc.

2.5.5 Interfaz gráfica de usuario

Ogre dispone de una librería para desarrollar la GUI de las aplicaciones muy versátil y potente llamada CEGUI

CEGUI es una biblioteca libre para la creación de ventanas y widgets para los motores donde la funcionalidad no está disponible de forma nativa. La biblioteca está orientada a objetos, escrito en C + +, y está dirigida a los desarrolladores de aplicaciones que deberían gastar su tiempo en crear grandes juegos y no la construcción de sub-sistemas de interfaz gráfica de usuario. Hasta la versión 0.4.1, CEGUI está bajo LGPL. Sin embargo, de 0,5 en la biblioteca se distribuye bajo la licencia MIT, que es menos restrictiva que LGPL.

2.5.6 Diseño de escenarios y personajes

Cualquier aplicación 3D necesita objeto.

Gracias al programa Blender (Chronister, 2006) es posible crear modelos 3D bajo licencia GPL. Ogre dispone de una herramienta que se instala en Blender para poder exportar los modelos de blender al formato de Ogre, concretamente .mesh. A su vez también se pueden crear animaciones de los objetos 3D, y a la hora de exportar, además del .mesh, se exportará también un archivo con la extensión .skeleton.

2.5.7 Resumen del capitulo

Se han visto diferentes alternativas para la creación de entornos virtuales, usando desde motores comerciales, a motores libres, como es el caso de ogre3D que es el elegido para el proyecto. Una vez elegido el motor grafico, se compararon también las librerías disponibles para este motor grafico y así completar el motor grafico.

Una de las necesidades del proyecto era que todas las herramientas que se usasen para desarrollar el proyecto fuesen libres, por ello la elección final fue:

- Ogre3D como motor grafico
- PyOgreOde de motor de física
- Socket para crear la red
- OgreAL como librería de sonidos
- CEGUI para crear la interfaz de usuario
- Blender para crear los modelos de personaje

Capítulo 3:
Un motor3D de entornos virtuales para evacuación

3 Un motor 3D de entornos virtuales de evacuación

Para crear un entorno virtual se necesita configurar el motor grafico, así como las librerías que se usaran para complementar las funciones requeridas por el proyecto.

En este capítulo tras explicar las necesidades del proyecto y mostrar la arquitectura general, se describirá el motor ogre3d para así familiarizarse con sus tecnicismos, así como una explicación de las herramientas que se han usado para facilitar el desarrollo de la aplicación.

En la segunda parte del capítulo se entrara en detalles de la implementación de cada uno de los prototipos, para finalizar en una explicación más detallada de la versión final del proyecto.

3.1 Especificación

La aplicación consiste en un simulador 3D en el que se pueden practicar y probar protocolos de evacuación en caso de que ocurriese cualquier tipo de contingencia, como por ejemplo, un incendio. En este caso en particular, la zona de actuación sería nuestra facultad de Informática de la Universidad Complutense de Madrid. Este simulador esta dentro del proyecto NOVA (Navegación basada en Ontologías mediante Verbalización de mensajes de Ayuda). Este simulador servirá primeramente para recoger información acerca de distintos usuarios creando logs de las conversaciones recogidas entre los alumnos y los guías. Tras estudiar y procesar esta información el entorno 3D servirá para poner en práctica la simulación de los sistemas inteligentes automáticos de guía y gestión de ayuda a usuarios que ha desarrollado el proyecto NOVA.

El objetivo de este proyecto es obtener un simulador multiusuario en el que exista una comunicación entre dos tipos de usuarios, alumno y guía, y poder guardar esas conversaciones para después procesarlas y poder crear sistemas inteligentes automáticos de guías y gestión de ayuda.

Para ello se requiere primeramente tener modelos 3D de los entornos en los cuales se quiera recoger los datos para generar los logs. En este proyecto el entorno se mantendrá fijo y será un modelo 3D de la Facultad de Informática de la Universidad Complutense de Madrid. El simulador tiene que poder usar estos entornos manera transparente para el usuario, de manera que el simulador funcione independientemente del entorno elegido para la simulación.

Además del entorno se necesitaran personajes que interactúen por él. Para ello el proyecto NOVA dispone de un generador de personajes, aunque para este proyecto no se usa. En su lugar se necesitan algún modelo 3D de un alumno para que haga la función de personaje. Al igual que con el entorno, el funcionamiento del simulador 3D no debe variar en función del modelo de personaje elegido.

Estos personajes pueden interactuar entre sí y con el entorno. Para ello se dota al mundo de una física, haciendo así que sea lo más real posible. Los alumnos podrán chocarse entre y con las paredes, impidiendo así que se traspasen los objetos.

El simulador contara con dos modos de uso diferente: Alumno y Guía. En el

modo alumno, el usuario manejara a un alumno por el entorno, en este caso la Facultad de Informática. El usuario tendrá libertad de movimientos, y podrá dirigirse al sitio que quiera, mientras espera que un guía se conecte y le elija. El alumno podrá tener uno o más guías, y deberá pedir información a estos para llegar a su destino, ya que se supone que este no conoce el lugar en el que se encuentra, ni el protocolo de evacuación. Si tiene más de un guía es decisión del usuario-alumno elegir a quien hace caso. Si se elige el modo guía el usuario deberá escoger un alumno de una lista compuesta por todos los usuarios que han entrado en modo alumno. Una vez elegido al alumno, el guía no dispondrá de un modelo de personaje, si no que podrá elegir entre dos tipos de cámaras para visualizar la escena. Tendrá la opción de ver todo desde la perspectiva del alumno al que esta guiando, o por el contrario podrá mover la cámara libremente por el entorno para poder disponer de más información de todo lo que rodea al alumno guiado.

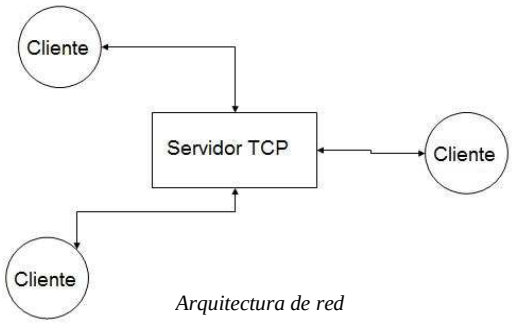
La comunicación entre el cliente y los guías puede realizarse mediante elementos visuales o a través del chat que se incluye en la interfaz grafica de ambas. Ambas interfaces, alumno y guía, cuentan con un mapa en el que un punto rojo representa al personaje que maneja el usuario-alumno y que guía el usuario-guía. El punto rojo se actualiza en el mismo momento que el alumno se mueve, de este modo si un guía ha optado por la cámara libre, puede conocer también la posición del alumno al que guía. El alumno podrá elegir a través del mapa, pinchando concretamente con el ratón sobre el mapa, el sitio al que quiere dirigirse y mandarle esta información al guía. Por su parte el guía pinchando en el mapa podrá colocar elementos en el entorno 3D del alumno al que guía de manera, si pincha en el punto (x,y) del mapa, en el punto (x,y,z), siendo z el nivel en el que se encuentra el alumno, aparecerá una flecha que indicara la dirección a la que debe dirigirse el alumno.

Si por el contrario se decide usar el chat para la comunicación el alumno podrá pedir o dar información al guía que le está dirigiendo (si hay más de un guía se enviara el mensaje a todos los guías). Los guías podrán usar el chat también para comunicarse con el usuario. Tanto el alumno como los guías verán los mensajes enviados entre los alumno-guía/s que están conectados entre sí.

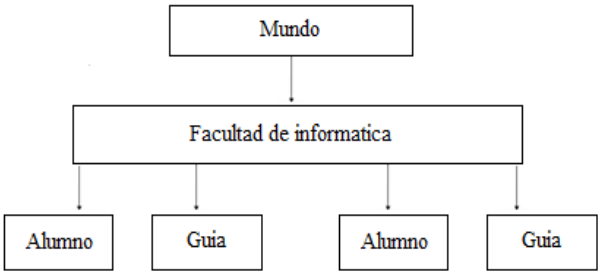
Para hacer posible la relación y comunicación entre los distintos usuarios que se encuentran en el entorno, es necesario contar con una red. La red se compondrá de un servidor central, al cual se conectaran todos los usuarios. Cada usuario será atendido por el servidor como si fuera un cliente, y por lo tanto para el servidor es indistinto si el cliente es un alumno o un guía.

3.2 Arquitectura y diseño

La arquitectura de la aplicación se puede dividir en dos subarquitecturas, una referida al modelo de comunicación entre los distintos clientes, y otra interna de cada cliente



Arquitectura de red



Arquitectura de cada cliente

El modelo cliente-servidor es el que mejor cubre las necesidades del proyecto desarrollado, ya que la forma en la que se van a relacionar los usuarios entre sí y con los guías no va a ser una conexión directa, todas las peticiones serán enviadas al servidor, las ventajas residen en la organización conseguida gracias a la centralización de la gestión de la información y la separación de responsabilidades, todo ello contribuye a que el diseño sea muy claro y sencillo.

Los clientes serán de dos tipos, por un lado un tipo de cliente denominado Alumno, que es el que se va a mover libremente por el entorno virtual y el que precisara la ayuda del otro tipo de cliente que se denomina Guía -El guía podrá circular libremente.

Por el entorno virtual y ser capaz de ver a todos los usuarios que haya en un determinado momento. La relación entre ambos es la siguiente: un guía solo puede tener un jugador al que guiar, pero un jugador puede tener varios guías para que le guíen, será decisión suya a quien seguir. Este tipo de relación ha sido una decisión de implementación que jugará un papel importante a la hora de recoger los datos generados por el experimento.

La interacción entre ambos tipos de clientes será el de un sistema de comunicación basado en el paso de mensajes, donde el jugador podrá indicar un sitio al que le gustaría ir, ya que se puede tratar de alguien ajeno a la facultad y no conocerla de antemano, o por otro lado podría necesitar pedir algún tipo de vía de escape en caso de un incidente. Esta comunicación se realizara a través de un chat, en el cual los usuarios escriben haciendo sus peticiones en el caso de ser un alumno, o dando soluciones si es el guía.

Otra forma de comunicarse es mediante elementos visuales, como por ejemplo pedir ayuda si eres un Alumno haciendo clic con el ratón en el mapa que dispone, o guiando mediante flechas que colocara el guía también usando el ratón.

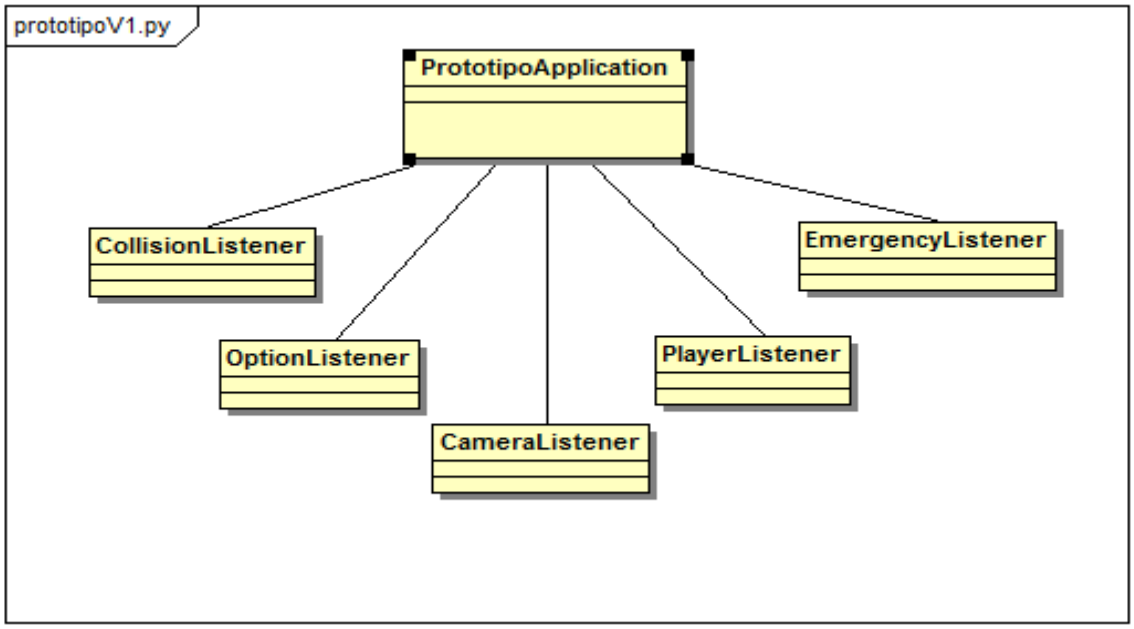
La finalidad última de la aplicación es la de recoger todas esas conversaciones en algún tipo de sistema de almacenamiento, como por ejemplo un "log", para que después se analice esa información recopilada, y se pueda ver como es la forma que tiene la gente de comunicarse y encontrar patrones en dichas conversaciones con el fin de dotar al guía de una inteligencia artificial que permita crear dichas indicaciones de forma automática.

3.2.1 Ficheros fuente

Los tres prototipos están basados en esta arquitectura, pero cada prototipo por su parte la implementa de un modo diferente, evolucionando siempre a mejor, mejorando así la claridad del código y el rendimiento de la aplicación. Mientras que en el prototipo V1 todo el código se encuentra en un solo fichero fuente, a partir del prototipo V2 el código se reparte entre diferentes ficheros. Los tres prototipos tienen un fichero resources.cfg en el cual se especifican la localización de las fuentes externas como puede ser los modelos o sonidos. Además cada prototipo cuenta con su propio Api, para facilitar el desarrollo de las futuras ampliaciones.

3.2.1.1 Primer prototipo

El primer prototipo solo cuenta con la arquitectura del cliente, puesto que no existe comunicación de red. Tiene un único fichero fuente (prototipo1.py) en el cual están todos los métodos que requiere el prototipo para su funcionamiento. Cuenta con una clase principal (PrototipoApplication). En esta clase se cargan todos los modelos y sonidos que se usan para crear el mundo de la aplicación. Se inicializan todas las variables y se llaman a los diferentes frameListener. Los frame Listener son clases que tienen dos métodos especiales, frameStarted y frameEnded. El primero ejecuta su contenido al principio de cada frame, y el segundo hace lo mismo con su contenido pero al final del frame. Los framListener son además clases. En este prototipo están el collissionListener, para detectar las colisiones entre objetos, opcionesListener que se encarga de modificar las opciones de la aplicación, CameraListener para cambiar el tipo de cámara que se usa, PlayerListener es el encargado de crear al alumno, y además se encarga de capturar los eventos para su movimiento y para acabar un EmergencyListener que maneja el modo emergencia. Todos estos modos se explicaran con más detalles en la sección prototipos.



3.2.1.2 Segundo prototipo

En este prototipo se incluye también la arquitectura de red. Debido a esto se crea un nuevo modulo llamado Client.py el cual contiene la clase ClientApplication que tiene una serie de métodos para comunicarse con el servidor y con otros clientes. Además a la hora de crearse lanza un hilo el cual está constantemente a la espera de nuevos mensajes (espera activa).

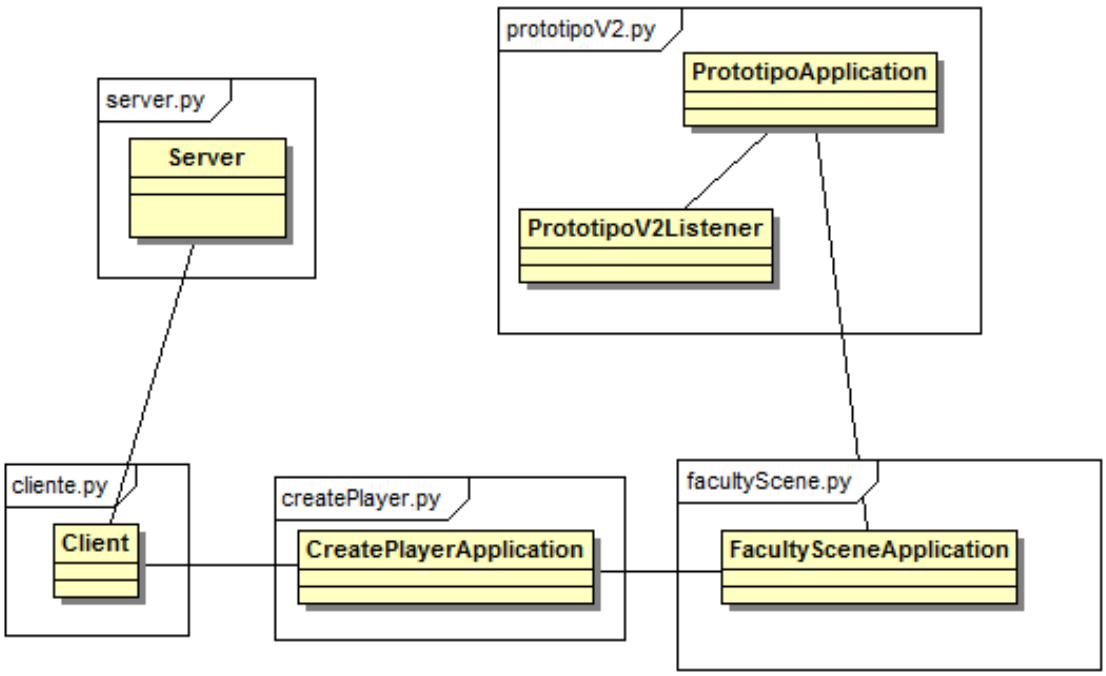
Para atender a los clientes se crea modulo llamado Server.py. Este modulo cuenta con una clase ServerApplication que crea un servidor socket UDP. Esta clase esta siempre a la espera de nuevos clientes. Cuando se acepta la conexión de un nuevo cliente se crean dos hilos asociados a este cliente. Uno sirve para mandarles mensajes desde el servidor, y otro para recibir mensajes de ese cliente.

El modulo prototipoV1 se ha divido en este prototipo creando nuevos módulos, y juntando clases. De este modo el código queda más claro y además será más fácil entenderlo y así poder hacer futuras ampliaciones.

El modulo PrototipoV2.py cuenta con una clase PrototipoApplication que se encarga de inicializar el "mundo" sobre el que se va a desarrollar la aplicación, crea la física de ese mundo y se encarga además de recoger la entrada de eventos del teclado. Crea la clase PrototipoV2listener que es el encargado de llamar a los métodos frameStarted y frameEnded de los demás módulos.

El modulo FacultyScene.py es el encargado de cargar el modelo de la facultad y establecer la física del edificio. Está compuesto de la clase FacultySceneApplication que realiza lo dicho anteriormente.

Para crear los diferentes personajes se utiliza la clase CreatePlayerApplication que se encuentra en el modulo CreatePlayer.py. Además esta clase recibe los eventos recogidos en el modulo PrototipoV2 y realiza la acción correspondiente a cada evento.



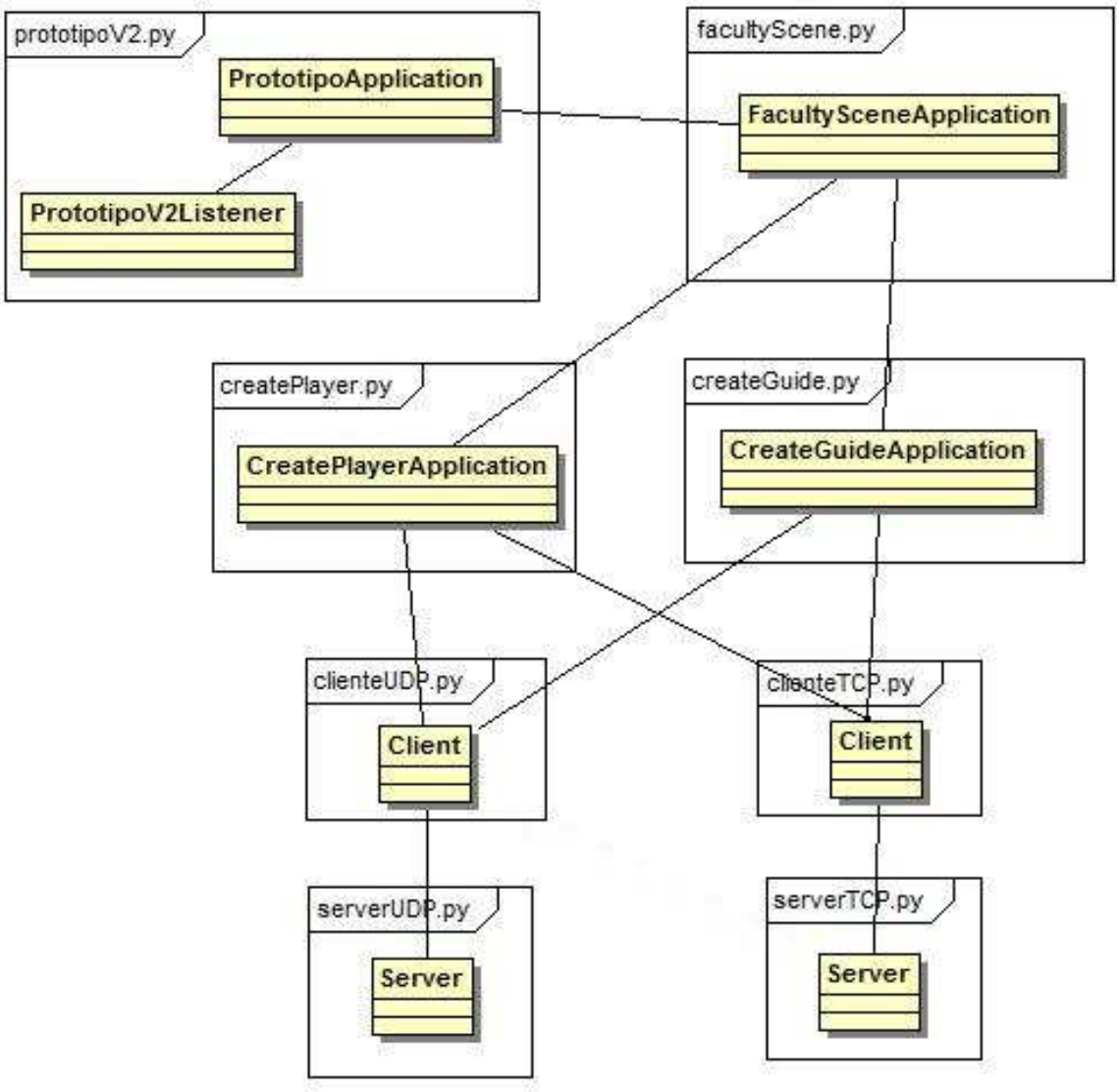
3.2.1.3 Tercer prototipo

Basado completamente en el segundo prototipo, los módulos existentes continúan teniendo las mismas funcionalidades. Se introduce un nuevo modulo serverTCP.py que añade al prototipo la posibilidad de conexión TCP. Este modulo tiene la misma implementación y funcionamiento que server.py del prototipo anterior, salvo a la hora de crear el socket, que en este caso creara uno de tipo TCP.

Para conectarse mediante TCP se crea un nuevo modulo clientTCP.py que es capaz de realizar una conexión TCP entre el cliente y el nuevo servidor. Este modulo cuenta con la clase ClientApplication que crea la conexión TCP. A diferencia del ClientApplication del prototipo anterior en esta clase los métodos que hay son solo para comunicarse con el servidor TCP. Aparte de esto también lanza un hilo como su homónimo UDP.

Por su parte los módulos cliente.py y server.py han sido renombrados a clienteUDP.py y serverUDP.py respectivamente.

Se añade un nuevo modulo llamado guide.py, que es el encargado de inicializar y crear todo lo referente al nuevo guía que se introduce en este prototipo. La clase que realiza estas acciones es la llamada CreateGuideApplication.



3.3 Herramientas

Para facilitar la creación de la aplicación y mejorar la productividad se ha hecho uso de diferentes herramientas para el desarrollo de la aplicación.

Todo el software con el que se ha desarrollado el proyecto es libre. El software libre (Abella, 2004)proporciona muchas ventajas a la hora de crear un proyecto.

Es muy económico, el coste de usar estas herramientas es cero. Existen herramientas libres que pueden tener una tarifa de distribución, pero no es el caso de las elegidas para este proyecto.

La calidad de este software no tiene que envidiar nada al software de pago alternativo que existe, siendo en muchos casos superior, debido a que gracias a las distintas distribuciones que se hacen de un mismo software existan diferentes versiones cada una acorde a un tipo de usuario.

El software libre tiene un soporte y una compatibilidad a largo plazo. Este punto, más que una ventaja del software libre, es una desventaja del software propietario, puesto que el software propietario, una vez alcanzado sus objetivos, no interesa seguir siendo desarrollado.

Tienen formatos estándar, con lo cual se consigue una interoperatividad más alta entre sistemas, evitando así incompatibilidades.

Al ser código abierto, todo el mundo puede auditarlos, y así tener sistemas muchos más seguros.

Existe una comunidad muy extensa desarrollando todo tipo de software libre, de manera que pueden solucionar los problemas de un programa rápidamente.

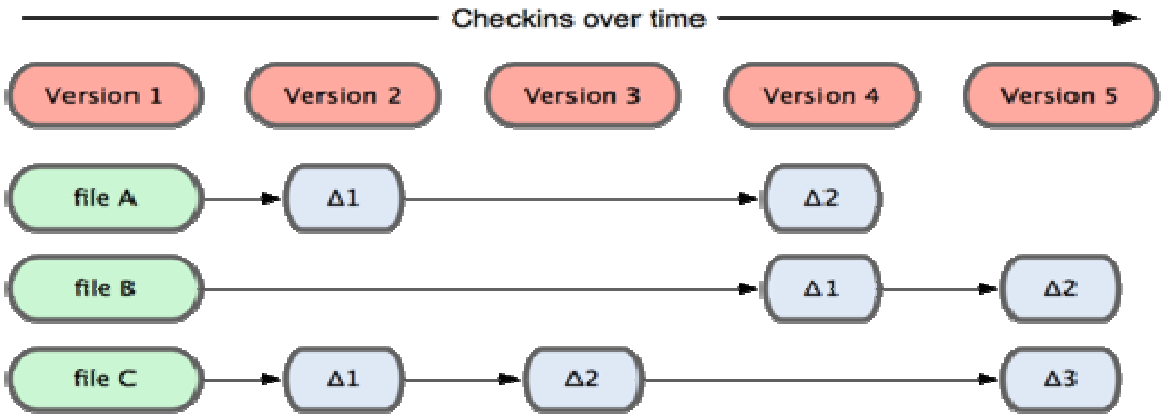
3.3.1 Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar aplicaciones. Dispone de numerosos módulos con los cuales se le pueden integrar funcionalidades como por ejemplo PyDev, que sirve para desarrollar aplicaciones escritas en Python. Al ser código libre existe una gran documentación sobre este programa, y de este modo escribir código con él hace que aumente la productividad.

3.3.2 Control de código

Para poder trabajar correctamente modificando partes especificas del código, y tener un control de versiones, se puede usar GIT. Git (Swicegood, 2009) (Burgess, 2010) fue diseñado por Linus Torvalds. En un principio se ideo GIT para ser un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario. Git es usado por proyectos de mucha relevancia como el grupo de programación del núcleo Linux.

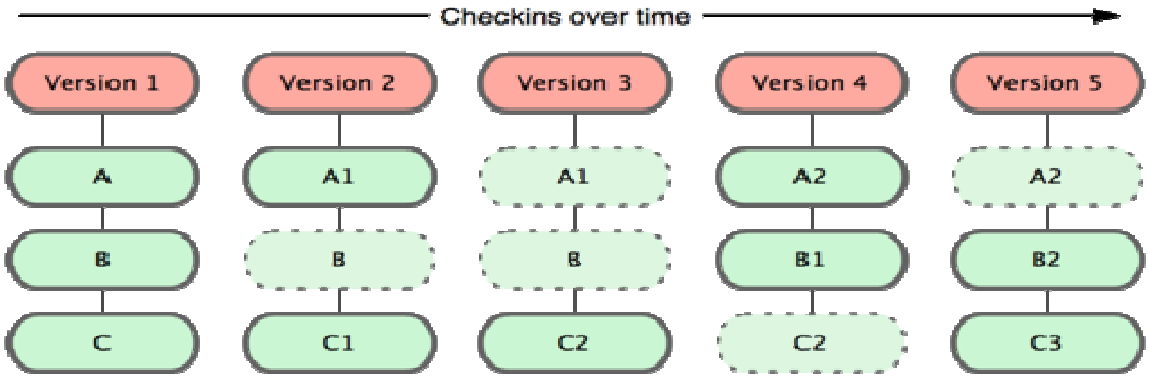
La principal diferencia entre Git y cualquier otro VCS (Subversión y compañía incluidas) es cómo Git modela sus datos. Conceptualmente, la mayoría de los demás sistemas almacenan la información como una lista de cambios en los archivos. Estos sistemas (CVS, Subversión, Perforce, Bazaar, etc.) modelan la información que almacenan como un conjunto de archivos y las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo.



Otros sistemas tienden a almacenar los datos como cambios de cada archivo respecto a una versión base.

.

Git no modela ni almacena sus datos de este modo. En cambio, Git modela sus datos más como un conjunto de instantáneas de un mini sistema de archivos. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente hace una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, Git no almacena el archivo de nuevo, sólo un enlace al archivo anterior idéntico que ya tiene almacenado.



Git almacena la información como instantáneas del proyecto a lo largo del tiempo

Esta es una distinción importante entre Git y prácticamente todos los demás VCSs. Hace que Git reconsidere casi todos los aspectos del control de versiones que muchos de los demás sistemas copiaron de la generación anterior. Esto hace a Git más como un mini sistema de archivos con algunas herramientas tremendamente potentes construidas sobre él, más que como un VCS.

La mayoría de las operaciones en Git sólo necesitan archivos y recursos locales para operar. Como toda la historia del proyecto está en el disco local, la mayoría de las operaciones parecen prácticamente inmediatas.

Todo en Git es verificado mediante una suma de comprobación antes de ser almacenado, y es identificado a partir de ese momento mediante dicha suma (checksum en inglés). Esto significa que es imposible cambiar los contenidos de cualquier archivo o directorio sin que Git lo sepa. Esta funcionalidad está integrada en Git al más bajo nivel y es parte integral de su filosofía. No puedes perder información durante su transmisión o sufrir corrupción de archivos sin que Git sea capaz de detectarlo.

El mecanismo que usa Git para generar esta suma de comprobación se conoce como hash SHA-1. Se trata de una cadena de 40 caracteres hexadecimales (0-9 y a-f), y se calcula en base a los contenidos del archivo o estructura de directorios en Git.

```
24b9da6552252987aa493b52f8696cd6d3b00373
```

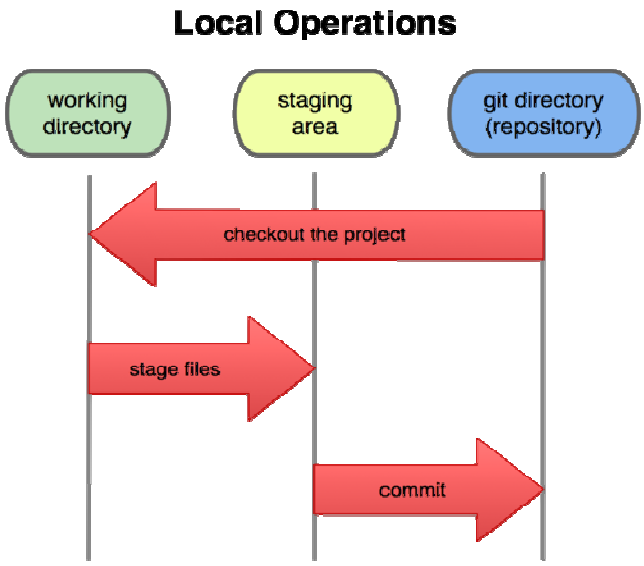
Ejemplo de hash SHA-1

Cuando se realizan acciones en Git, casi todas ellas sólo añaden información a la base de datos de Git. Es muy difícil conseguir que el sistema haga algo que no se pueda deshacer, o que de algún modo borre información. Como en cualquier VCS, se puede perder o estropear cambios que no se han confirmado todavía; pero después de confirmar una instantánea en Git, es muy difícil de perder, especialmente si se envía (push) a la base de datos a otro repositorio con regularidad.

Esto hace que usar Git sea un placer, porque se sabe que se puede experimentar sin peligro de fastidiar gravemente las cosas.

Git tiene tres estados principales en los que se pueden encontrar los archivos: confirmado (committed), modificado (modified), y preparado (staged). Confirmado significa que los datos están almacenados de manera segura en la base de datos local. Modificado significa que se ha modificado el archivo pero todavía no se ha confirmado a la base de datos. Preparado significa que se ha marcado un archivo modificado en su versión actual para que vaya en la próxima confirmación.

Un proyecto de Git tiene tres secciones principales: el directorio de Git (Git directory), el directorio de trabajo (working directory), y el área de preparación (staging área).



Directorio de trabajo, área de preparación, y directorio de Git.

El directorio de trabajo es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que se puedan usar o modificar.

El área de preparación es un sencillo archivo, generalmente contenido en el directorio de Git, que almacena información acerca de lo que va a ir en la próxima confirmación. A veces se denomina el índice, pero se está convirtiendo en estándar el referirse a ello como el área de preparación.

El flujo de trabajo básico en Git seria:

- Modificación de una serie de archivos en el directorio de trabajo.
- Se preparan los archivos, añadiendo instantáneas de ellos al área de preparación.
- Confirmación de los cambios, lo que toma los archivos tal y como están en el área de preparación, y almacena esa instantánea de manera permanente en el directorio de Git.

Si una versión concreta de un archivo está en el directorio de Git, se considera confirmada (committed). Si ha sufrido cambios desde que se obtuvo del repositorio, pero ha sido añadida al área de preparación, está preparada (staged). Y si ha sufrido cambios desde que se obtuvo del repositorio, pero no se ha preparado, está modificada (modified).

Aunque parezca complicado el uso GIT, su manejo se vuelve trivial gracias a TortoiseGit. TortoiseGit es una interfaz grafica para controlar GIT muy cómoda para el usuario. Dispone de indicadores visuales para ver el estado de los ficheros y carpetas del proyecto.

3.3.3 Berlios

La funcionalidad de Berlios es apoyar los desarrollos de software libre (Open Source) mediante servicios en línea como sistemas de control de versiones (GIT), listas de correo, sistemas de control de erratas y otras funcionalidades. BerliOS pretende ayudar en todos los aspectos del desarrollo de software libre (Open Source).

El portal principal para llevar esto a cabo es Developer. Developer está basado en la versión 2.5 de SourceForge y provee servicios como la gestión de versiones, listas de correo, sistema de control de erratas, documentación y parches, etc. etc.

La comunicación entre desarrolladores y entre desarrolladores y patrocinadores es otra área donde BerliOS quiere aportar soluciones. Éstas incluyen portales Wiki y foros, además de listas de correo y direcciones de correo electrónico personales.

3.3.4 Diseño de escenarios y personajes

Gracias al programa Blender (Chronister, 2006) se ha creado un personaje que hace a su vez de alumno de la universidad. Las animaciones también se han creado a través de Blender, y a la hora de exportar, además del .mesh el .skeleton.

Además del alumno, también se modelo para el primer prototipo un modelo que recreaba un recinto con una serie de habitaciones y las flechas indicativas que aparecen en la versión final.

Modelo creado en Blender de un alumno

3.4 Descripción del motor gráfico

El objetivo en un motor de renderizado es proveer una manera fácil y rápida de dibujar la escena en pantalla, ahorrándose así muchos de los engorrosos pasos de configuración de una API de más bajo nivel como puede ser DirectX u OpenGL. Gracias a la estructura de OGRE (Kerger, 2010), esto resulta bastante sencillo. Para comenzar, se pueden distinguir tres elementos básicos: las Entities, los SceneNodes y los SceneManagers.

Las **Entities** o entidades representan todos los objetos que pueden dibujarse en la escena. Se puede pensar en una entidad como cualquier cosa que es representada por una malla 3D. Un robot sería una entidad, un pez sería una entidad, el terreno sobre el que se mueven los personajes también sería una gran entidad. Sin embargo, otros objetos, como las cámaras o las luces, no son entidades. También cabría mencionar que la orientación y la posición no son propiedades propias de las entidades. Esto significa que no se puede poner directamente una entidad en una escena. En lugar de ello, hay que adjuntar la entidad a otra estructura llamada SceneNode, la cual contiene toda la información sobre la ubicación y orientación de la entidad.

Como ya se ha mencionado, los **SceneNodes** realizan un seguimiento de la ubicación y orientación de todos los objetos que se le atribuyen. Al crear una entidad, no es añadida a la escena hasta que no se le adjunta a un SceneNode. Su función es ser un contenedor de entidades además de otros objetos como cámaras o luces. Además, se pueden crear jerarquías de nodos, es decir, los SceneNodes pueden ser padres de otros SceneNodes. Cabe mencionar que las posiciones de los SceneNodes son relativas a las de sus padres. De esta forma, si un nodo llamado NodoB, que tiene la posición (3,0,0) y su padre, llamado NodoA, que es el nodo raíz y, que, por lo tanto, tiene sus coordenadas en escala absoluta, tiene la posición (2,0,0), la ubicación absoluta del NodoB será la (5,0,0).

Por último, los **SceneManagers** son los encargados de seguirle la pista a todos los objetos que se dibujan por pantalla. Pueden verse también como los encargados de todo lo que tenga que ver con el dibujo de la escena. Los SceneManagers poseen el nodo raíz de la jerarquía de SceneNodes, de forma que, a partir de él, se puede llegar a cualquier Entity de la escena. Además, los SceneManagers se encargan también del dibujo del terreno, por lo que en él se pueden especificar las características del mundo que se dibujará. Por ejemplo, si se va a realizar un juego de interiores, el SceneManager para ellos es distinto que si se trata de uno de exteriores. De hecho, es posible utilizar varios manejadores de escena para un mismo proyecto.

Una **cámara** es un objeto especial que funciona de forma parecida a un SceneNode. Una cámara tiene funciones como setPosition, yaw, roll y pitch y se puede adjuntar a cualquier SceneNode. Al igual que cualquiera de estos últimos, la posición de una cámara es relativa a la de sus padres. Para todos los movimientos y rotaciones se puede considerar, básicamente, una cámara como si de un SceneNode se tratara. Una de las características propias de las cámaras es que sólo se puede utilizar una cámara a la vez. Es decir, no se puede crear una cámara para ver una parte de la escena, una segunda cámara para ver otra parte y, a continuación, permitir o inhabilitar las cámaras sobre la base de la escena que queremos mostrar. La forma de conseguir esta funcionalidad es crear SceneNodes que actúen como "titulares de la cámara". Estos SceneNodes se colocan en el lugar y momento en el que queremos colocar la cámara. Cuando sea el momento de cambiar la cámara, simplemente le asignamos un determinado SceneNode.

Cuando se comienza a hacer frente al uso de múltiples cámaras aparece el concepto de **Viewport**. Esta clase es muy útil en estas situaciones. Para ello, es importante entender cómo OGRE decide qué cámara va a usar cuando se renderiza la escena. Puede darse el caso que varios SceneManagers estén corriendo al mismo tiempo. También es posible dividir la pantalla en varias áreas y tener cámaras para cada una de las zonas de la pantalla (un ejemplo bastante ilustrativo puede ser una partida para 2 jugadores en un juego de consola). Para entender cómo OGRE realiza una escena, basta considerar estos tres constructores: la cámara, el SceneManager, y la RenderWindow. La RenderWindow es la ventana en la que todo se muestra. El objeto SceneManager crea las cámaras para ver la escena. Ahora, se debe decir a la RenderWindow que cámaras debe mostrar en la pantalla y en qué porción de la ventana debe renderizarlas. El área donde se le dice a la RenderWindow que debe mostrar la cámara es el Viewport. En la mayoría de los usos típicos de OGRE, por lo general, será necesaria una sola Cámara, y, por tanto, sólo se dispondrá de un Viewport.

Otro concepto muy necesario es el de **bounding box**. El bounding box es el cuadro delimitador de la Entity a la que va asociada. Hay varios tipos de bounding box, está el triangle Mesh, que es recubre todos los polígonos de la Entity mediante triángulos creando una cuadro delimitador similar a la forma de la Entity, hay otros estáticos con formas cubicas, esféricas o cilíndricas. Existe también un tipo de boundingBox conocido como terrain, que es el encargado de delimitar al terreno, su característica principal es que esta bounding box es inamovible, es decir, no se puede desplazar mediante colisiones.

3.5 Prototipos

3.5.1 Objetivos de los prototipos

Se decidió dividir el proyecto en tres prototipos, de manera que cada uno de ellos integraba una funcionalidad nueva, ampliando así a su predecesor. Además así se conseguía tener una versión final más robusta, puesto que cada prototipo era testeado como si fuese un proyecto final, y así el prototipo siguiente tenía una base firme.

El tercer prototipo incorpora ya todas las funciones del proyecto, salvo el modelo final de la facultad, cargando en su lugar un modelo previo, que consiste en la 3 planta del edificio. Este prototipo es en el que está basada la versión final.

3.5.2 Primer prototipo

El objetivo de este primer prototipo era cargar modelos 3D, y detectar las colisiones que se producían en el, para posteriormente poder interactuar con ellos. Para ello usando blender, se creó un primer entorno, que consistía en un conjunto de habitaciones, por las cuales un personaje podía moverse, interactuando con los objetos que en ella había, concretamente una serie de puertas que se podían abrir y cerrar, un barril, y una caja. El resultado fue mejor del esperado debido a que al final no solo se detectaban las colisiones, sino que también se producían eventos al producirse estas, debido a que también se integro el motor de físicas.

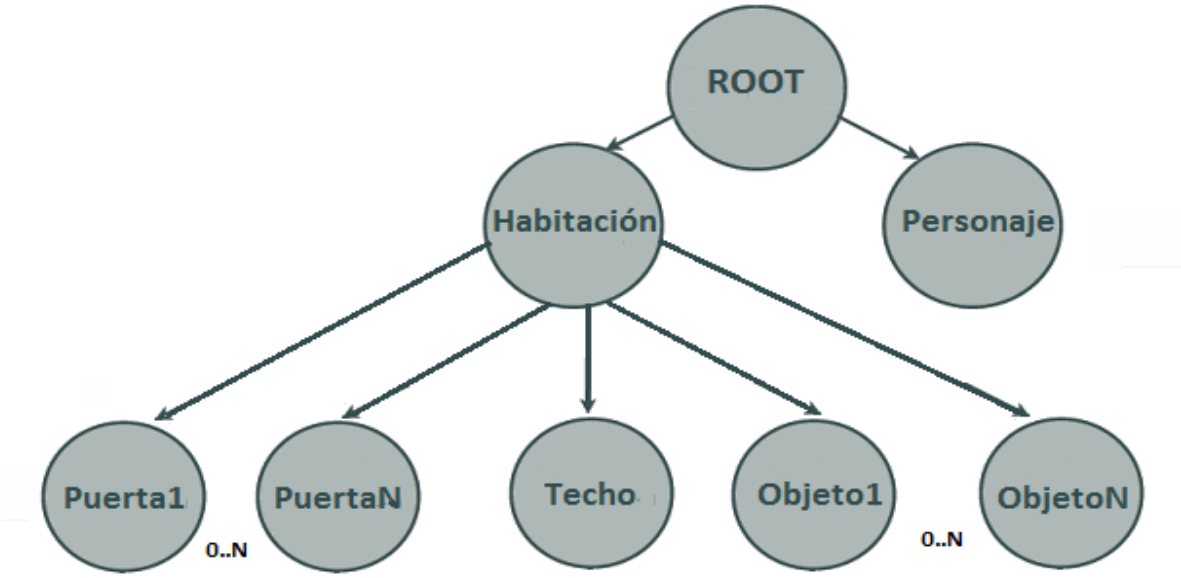
En la escena podemos observar un personaje que se controla con las teclas de dirección, y un escenario en el cual hay una serie de puertas que se pueden abrir para pasar a otras habitaciones, también hay bidones "muy pesados" que no se pueden mover, y cajas "menos pesadas" que pueden empujarse

3.5.2.1 Componentes:

El motor gráfico o de render En este primer prototipo se ha basado en librerías/APIs gráficas de OpenGL y DirectX, pudiendo elegir al inicializar la aplicación.

El grafo de escena. Del nodo principal cuelgan las dos Entities principales, la habitación y el personaje.

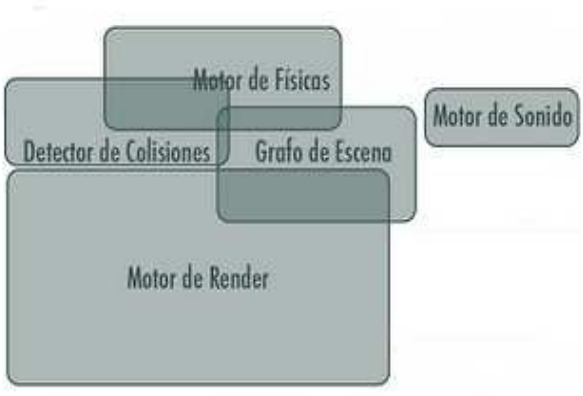
La habitación por su parte tiene diferentes objetos en su interior que van sujetos a su posición.



Tras probar los motores de física OgreBullet y OgreOde, los resultados obtenidos con OgreOde eran un poco superior, y como ambas librerías proporcionaban todos los requisitos necesarios para la aplicación, la elección es OgreOde.

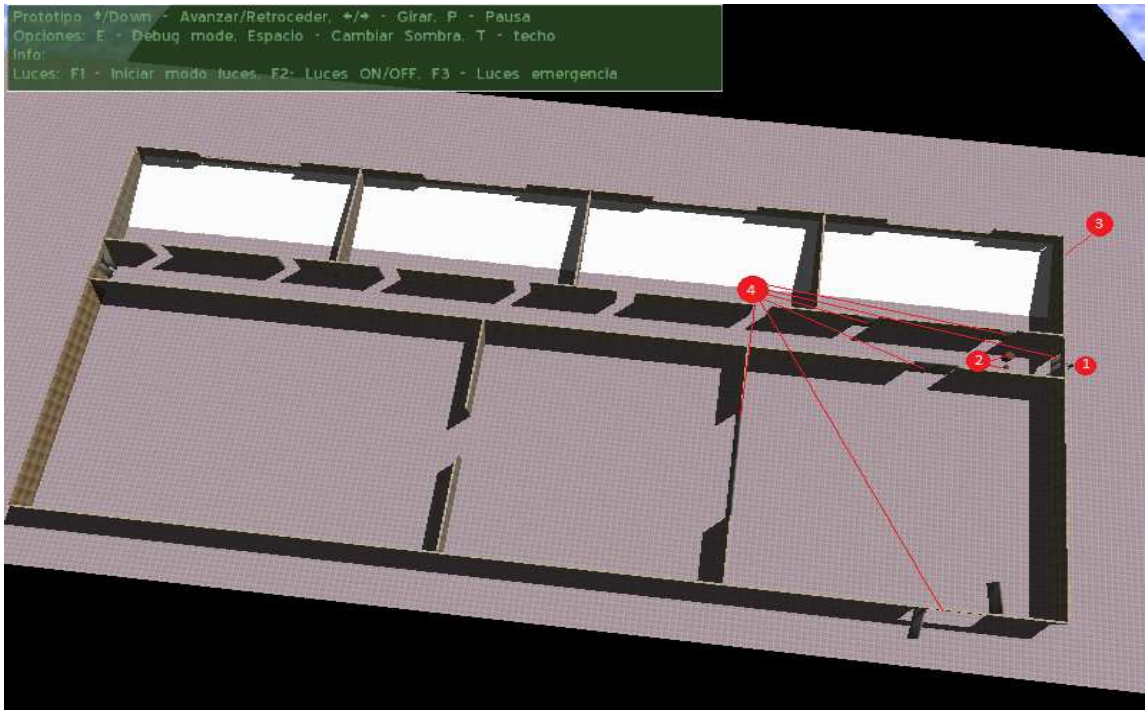
El detector de colisiones usa la intersección de los bounding box para saber si se ha producido choque o no. Los bounding box utilizados se comentaran cuando se hable sobre los objetos de la escena.

El motor de sonido que integramos es OgreAL, basado en OpenAl como se menciono anteriormente.



El SceneManager elegido es de tipo Interior, puesto que para esta aplicación es la que daba un mejor rendimiento debido a que todas las acciones ocurren en un espacio cerrado.

Con todo esto la escena quedaría del siguiente modo:



1-posición inicial del personaje

2- Objetos de la escena, una caja de poco peso que se puede mover, y un barril pesado que es imposible moverse

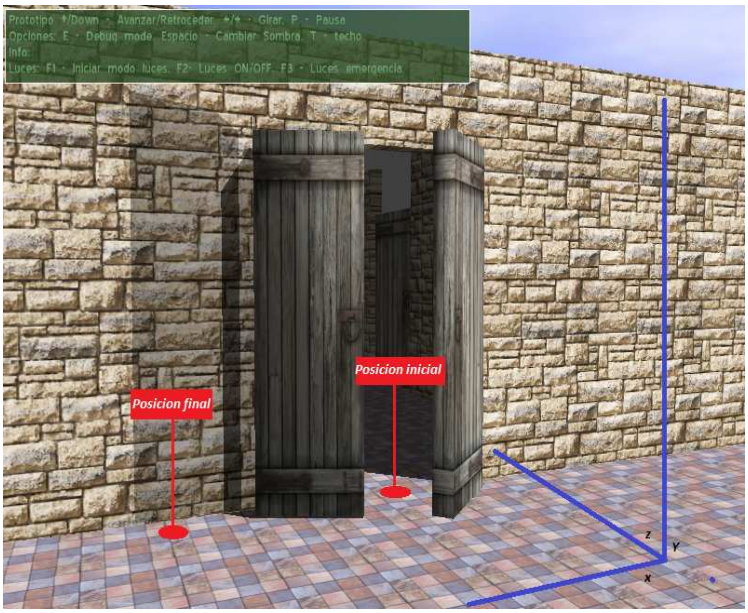
3-Paredes del escenario

4-Puertas

3.5.2.2 Objetos:

Habitaciones:

Mediante blender se ha creado un escenario que se compone de varias habitaciones. Las habitaciones tienen puertas que al empujarlas se abren, esto se ha conseguido creando joint en Ogre. El método joint de OgreOde establece su posición inicial y su posición final, además del plano sobre el que van a girar. Sería como recrear una bisagra.



Ejemplo de funcionamiento y posicionamiento de dos puertas

La boundingBox de la habitación es una "triangle Mesh", la cual crea miles de triángulos que rodean toda el área de la habitación, para poder detectar las colisiones que se producen con sus muros.

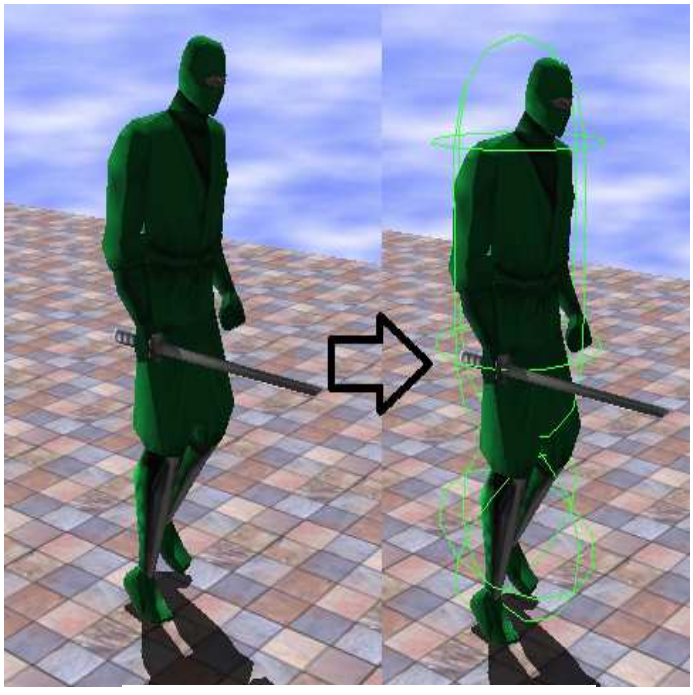
La boundingBox de las puertas es una "Single DinamicBox", que simplemente rodea a la puerta con un prisma rectangular de su mismo tamaño, y puesto que la puerta es un prisma rectangular es la mejor opción.

Bounding box que delimitan al escenario y a las puertas

Personaje: Para este prototipo se usa un personaje que viene con Ogre llamado "ninja". Este personaje dispone de varias animaciones, entre las que se encuentra la animación de caminar.

Para poder interactuar con el mundo debemos crearle un "Body"(cuerpo).Esto ya lo hemos hecho en el caso de la habitación y de las puertas, pero con el personaje será distinto porque queremos controlar también el movimiento de este por el escenario.

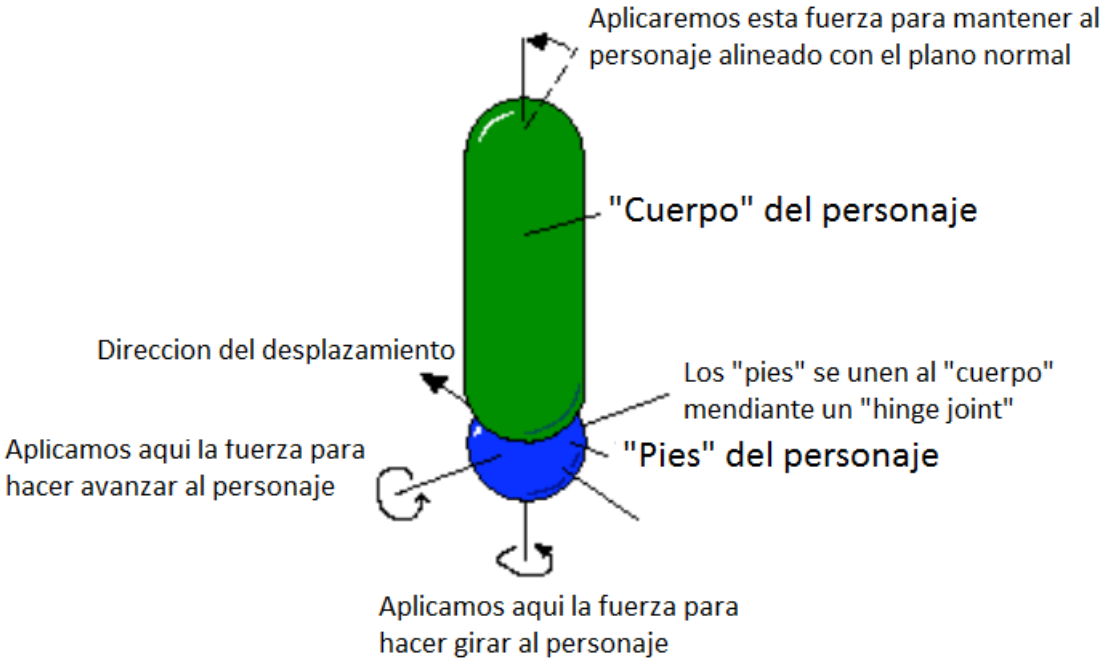
El Body se compone de dos boundingBox unidas (uno esférico y otro ovalado), desactivando las colisiones entre ellas.



En la foto se puede ver una Entity a la cual se le ha aplicado los boundingBox mencionados

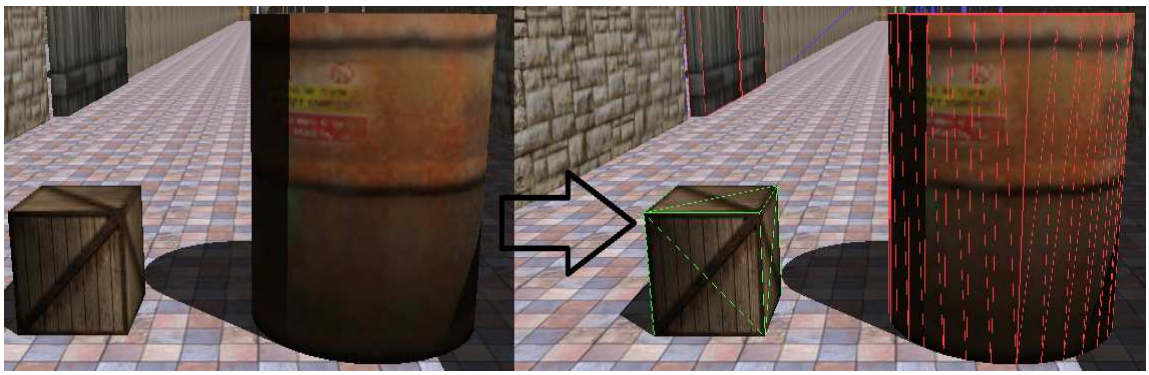
Una vez creados los dos cuerpos, necesitamos aplicarles las fuerzas para poder

hacer que el personaje se mueva, para ello se aplicaran las fuerzas del siguiente modo



Otros objetos. Para este prototipo se han introducido dos objetos más, un barril y una caja, ambos cogidos de los modelos que vienen con Ogre.

Los dos son recubiertos por una "Triangle Mesh". La diferencia entre ambos es el peso de cada uno. Mientras que la caja tiene un peso muy bajos, puede ser desplazado por el personaje, al barril se le ha asignado un peso elevado, por lo que el personaje no tiene fuerzas para desplazarlo. El peso se ha conseguido gracias al motor de físicas, al cual se le ha pasado como datos el peso del material correspondiente, que junto al volumen que ha calculado previamente al recubrirlo con el boundingBox, se obtiene el peso del objeto.



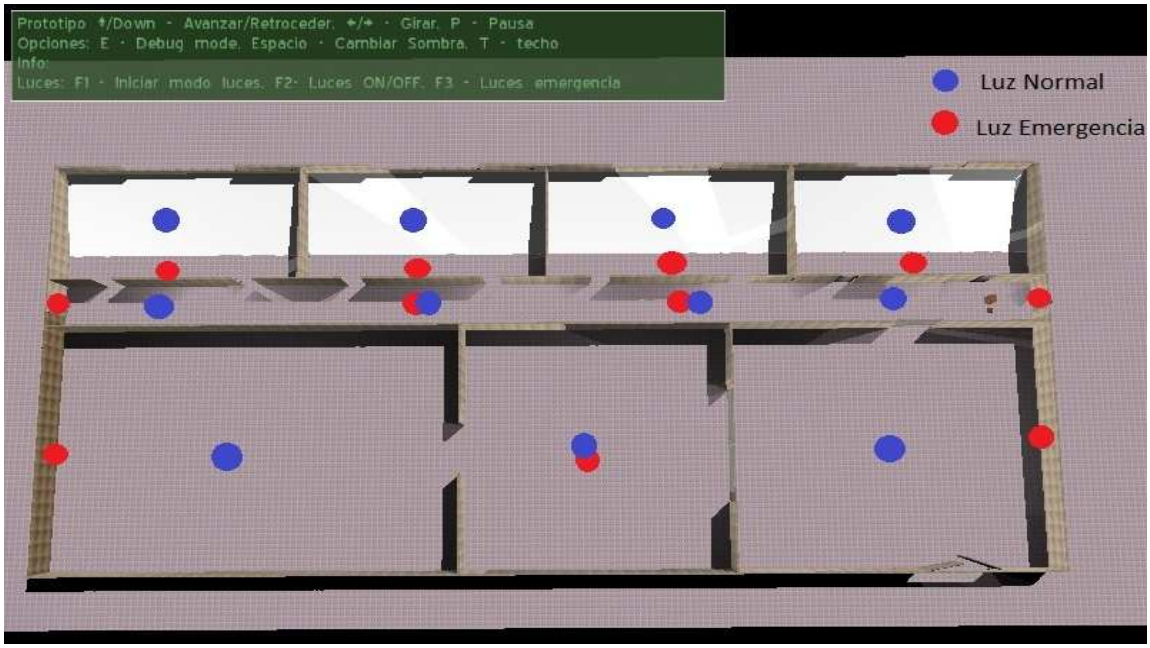
A la izquierda se vendos objetos del escenario. A la derecha se ven los mismos objetos recubiertos con un triangleMesh

3.5.2.3 Cámara

Se dispone de dos cámaras en este prototipo. Una libre con la cual se puede mover libremente y otra en tercera persona que va siguiendo al personaje. Para hacer posible la cámara en 3 persona, se va trasladando la cámara en cada frame según la posición del nodo del personaje. Previamente se ha configurado la distancia y altura a la que se encuentra respecto al personaje.

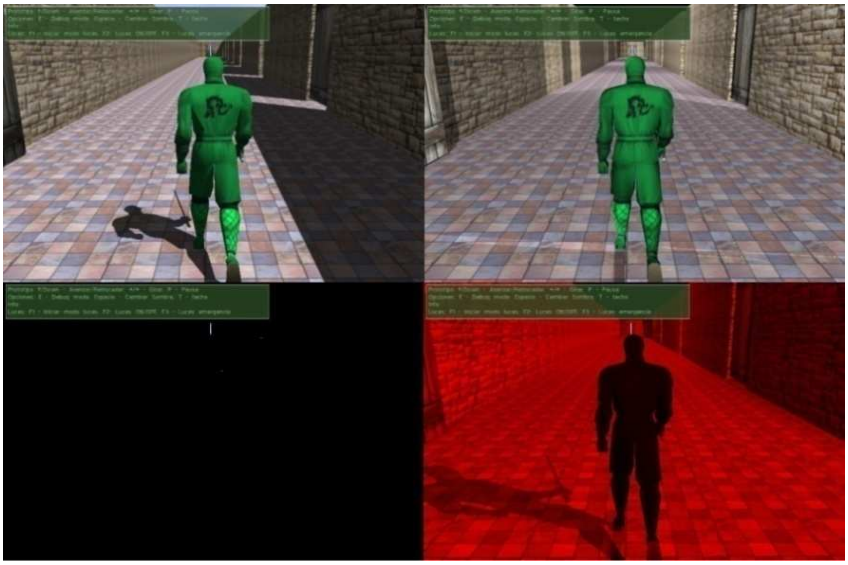
3.5.2.4 Modo Luces

Pensando en las mejoras que se pueden añadir a la versión final del proyecto se ha introducido un "modo luces". Lo que varía en este modo es que elimina la luz del SceneManager, se crea una luz artificial de tipo SPOTLIGHT que simula la luz solar. A su vez en el escenario se han situado varias luces a la altura del techo que hacen la función de los focos del edificio.



Situación de los tipos de luces en el edificio

Si la luz está encendida se verá una luz amarilla simulando la luz solar. También se pueden apagar las luces, estando todo a oscuras, y un tercer tipo de luz rojizo simula las luces de emergencia del edificio.



De izquierda a derecha y después segunda fila: Con luz en el SceneManager, Con luces artificiales, sin luces, con luces de emergencia

3.5.2.5 Entrada estándar

Para poder efectuar acciones se recogen a través eventos que se producen al pulsar las teclas, mediante un buffered input de la librería OIS para Ogre3D.

Se ha puesto una condición en todas las teclas menos en las referentes a la dirección. Esta condición es que solo se puede pulsar una tecla cada medio segundo, para que así no haya colapso, y se puedan registrar bien los eventos.

Resumen de teclas:

Teclas de dirección: Movimiento del personaje

P: Pausa el juego

T: Aparece/desaparece el techo

Espacio: cambia el tipo de sombra

E: "modo debug"

F1: "modo luces"

F2: luz on/off

F3: Luz emergencia

3.5.3 Segundo prototipo

El objetivo del segundo prototipo era conseguir una comunicación en red entre distintos clientes y establecer una entrada estándar de manera que las órdenes recibidas por los clientes sean indistintas del método por el cual se introducen dichas órdenes.

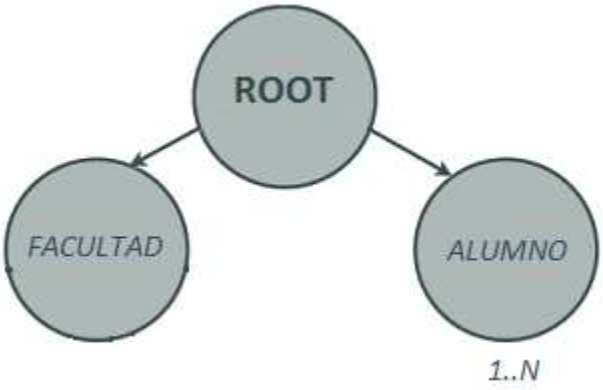
Además se ha introducido ya un modelo de la facultad, el cual dispone de dos plantas, y de un estudiante con movimiento que puede moverse por la facultad.

Ejemplo del prototipo 2, en el que se ve a un alumno caminando por la facultad

3.5.3.1 Componentes:

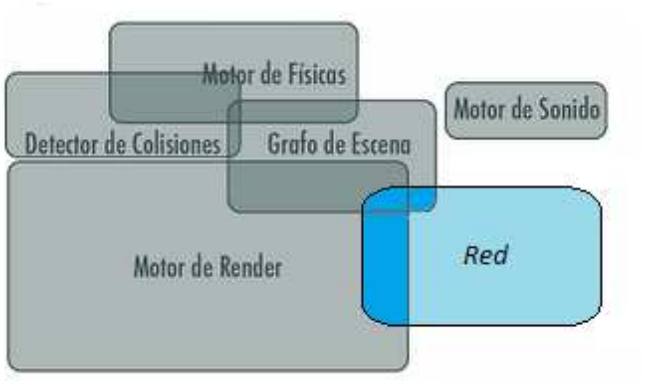
El motor gráfico sigue basándose en librerías/APIs gráficas de OpenGL y DirectX, pudiendo elegir al inicializar la aplicación.

El grafo de escena sigue la misma forma que en la primera versión. Un nodo Root del cual cuelga por una parte la facultad y por otra todos los personajes/clientes que hay en escena en cada momento.



Se sigue usando también OgreOde como motor de física y OgreAL como motor de sonido.

La novedad en este prototipo es la inclusión de la red. Para ellos se incluye el modulo socket con el cual realizamos una conexión de tipo UDP.



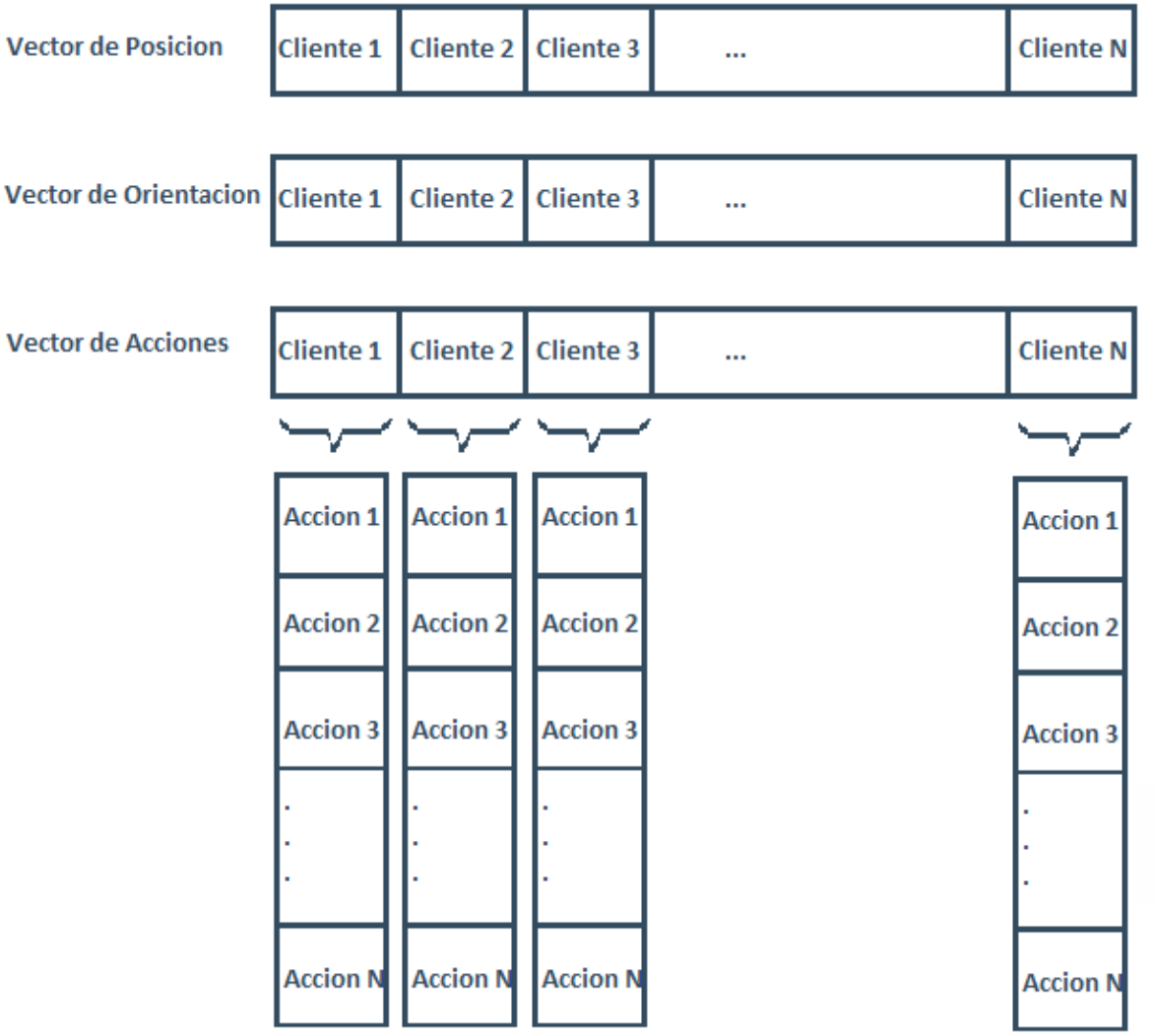
En azul se puede ver el nuevo modulo que se ha incluido

3.5.3.2 Red

La red está basada en el protocolo UDP, y sigue el siguiente esquema

- Apertura del servidor
- Apertura del prototipo y establecer una conexión
- Esperar a que el servidor acepte la conexión y mande una “ip” al cliente, una lista de los clientes conectados y la posición y orientación de cada uno de ellos.
- Se mandan las acciones a todos los clientes conectados y al servidor
- Vuelta a 4

Una vez que se han hecho estos pasos el personaje puede moverse por el mundo. Cada vez que un cliente realiza una acción se manda por broadcast a todos los demás clientes conectados y al servidor la posición y orientación que tiene tras efectuar esa acción, además de mandar también la propia la propia acción. Estos mensajes se mandan en el inicio de cada frame de los prototipos y los guarda cada cliente en sus respectivos vectores:



Contenido de los tres vectores que tiene cada cliente, para guardar la información de los demás clientes

Al finalizar cada frame se obtiene la última acción que ha realizado cada personaje y se ejecuta, sacando esa acción de la lista de acciones del cliente correspondiente.

Los mensajes que se mandan están compuestos por:

- Tipo : Actualmente solo se pueden mandar mensajes de tipo Broadcast
- Acción : Pudiendo ser orientación, posición o acción
- Posición: Indica el cliente que lo ha mandado para poder actualizar el valor en la lista de clientes
- Datos: La información que se quiere mandar
    - Si se manda la posición los datos son px (posición x), py (posición y) y pz (posición z)
    - Si se manda la orientación los datos son ow(orientación w), ox (orientación x), po (orientación y) y oz (orientación z)
    - Si se manda una acción se manda un único dato que es la acción que se va a realizar. Actualmente los tipos de acciones son:
        - Up
        - Down
        - Left
        - Right

Usando el módulo pickle de Python se serializan estos datos y se mandan a través de la red. Al recibirlo se hará la operación inversa y se obtendrán los datos correctos.

En estas imágenes se observar la visión que tiene cada prototipo de la red. En ambos están los dos personajes pero cada uno lo ve desde su perspectiva

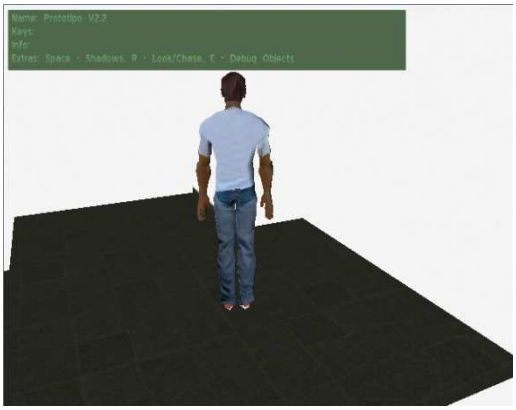
3.5.3.3 Sombras

En esta versión se ha añadido la posibilidad de cambiar el tipo de sombras.

Modulative Texture Shadows (SHADOWTYPE_TEXTURE_MODULATIVE) - La más baratas de usar de las tres. Crean un renderizado en blanco y negro en una textura de la entidad sombreada, y a continuación se aplica a la escena.

Modulative Stencil Shadows (SHADOWTYPE_STENCIL_MODULATIVE) - Esta técnica dibuja todos los volúmenes de sombras como una modulación después de que todos los objetos no transparentes hayan sido dibujados. No es tan cara como las Additive, pero tampoco tan precisa.

Additive Stencil Shadows (SHADOWTYPE_STENCIL_ADDITIVE) Esta técnica dibuja cada luz como una pasada disjunta en la escena. Esto es muy caro computacionalmente ya que cada luz requiere una pasada adicional de renderizado en la escena.



Modulative Texture

Modulative Stencil

Additive Stencil

3.5.3.4 Entrada estándar

Se sigue usando el modulo IOS de Ogre3D para recogerlos eventos que se producen en el teclado

Resumen de teclas:

F1 = Iniciar

UP y w = Avanzar

Down y s = retroceder

Iz y a = girar a la izquierda

De y d = girar a la derecha

P = Pausa

Espacio = Tipo de sombra

R = Tipo de visionado de malla (normal, polígono, puntos)

E = Cambiar entre modo Debug y normal.

3.5.4 Tercer prototipo

En este prototipo se puede elegir entre guiar a otro usuario a través de la facultad, o ser él el guiado. La comunicación entre el guía y el usuario se realiza a través de un chat que hay en pantalla. El usuario puede pedir ayuda escribiendo en él, o pinchando en un mapa que hay en pantalla, eligiendo así el sitio al que quiere ir. El guía por su parte ira poniendo marcas para que el usuario las siga y pueda llegar a su destino.

El primer cambio apreciable es la ventana que aparece nada más comenzar. En ella el usuario tendrá la opción de entrar como guía o usuario.

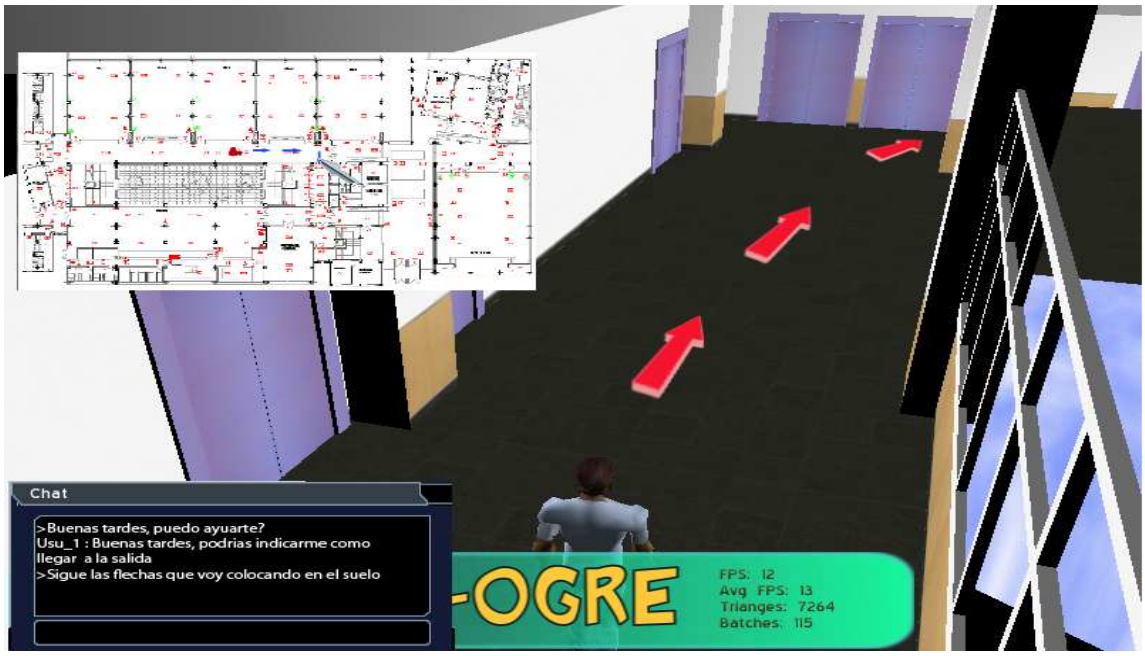
Si decides entrar como alumno, Podrás moverte por la facultad libremente, a la espera de que un guía se conecte, y así poder ayudarte a llegar a tu destino.

La interfaz gráfica del alumno ha cambiado ligeramente. Se ha incluido un chat para comunicarse con el guía, y un mapa, en el que se marca su posición, que se actualiza según se mueva este.

Interfaz grafica del alumno

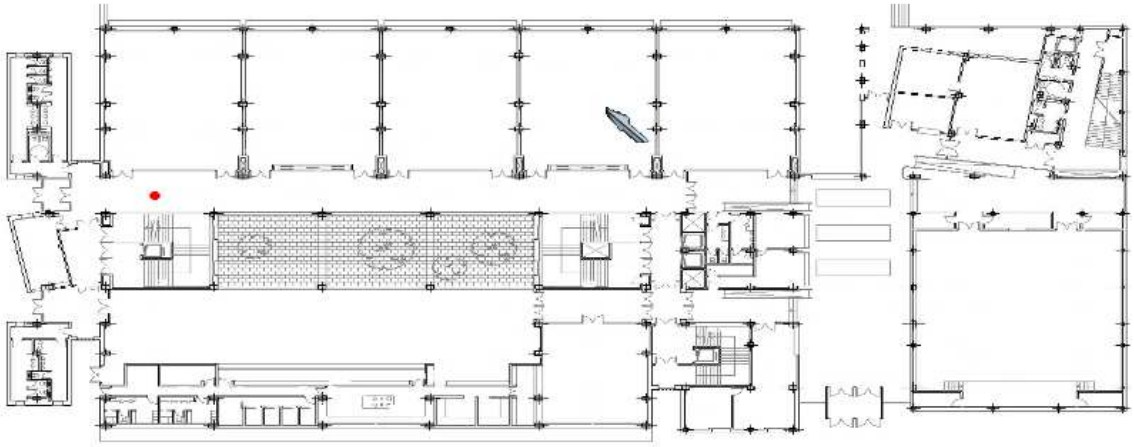
Si en lugar de alumno decides ser guía, se abrirá una ventana en la cual se mostraran todos los alumnos que hay conectados, y si tiene algún guía asociado. Un alumno puede tener varios guías, y el elegirá a quien hace caso, pero un guía no puede tener varios alumno.

Una vez elegido el alumno, la cámara se situara sobre este, pero el guía tendrá libertad para moverla. Podrá dar instrucciones al alumno a través del chat, o pinchando en el mapa, colocando así marcas en el modelo de la facultad que el alumno podrá seguir.



Interfaz grafica del guía

En la interfaz grafica tanto del alumno como del guía, aparece un mapa de la zona en la que se encuentra el alumno. Este mapa puede redimensionarse ocupando toda la pantalla, o aparece en tamaño más pequeño, para así poder ver el escenario. La funcionalidad del mapa es saber donde se encuentra el alumno. Para ello en el mapa se coloca un punto rojo que representa al alumno que se está manejando o guiando.



Mapa de la planta baja de la facultad, con el punto rojo que representa al alumno, que está situado en frente del aula 5.

3.5.4.1 Red

Al realizar las pruebas en el prototipo 2, se encontraron unos errores en la comunicación, de manera que si la red de comunicación no era buena algunos mensajes se perdían (UDP no garantiza la entrega de mensajes) y de ese modo había un desfase entre los distintos clientes.

Por ello en esta versión se dio la opción de usar conexión mediante TCP, que si garantiza la entrega de mensajes, pero consume también más recursos. El testeo mediante TCP fue muy positivo, los clientes no se desfasaban entre ellos. Además también se añadió una función al servidor, el cual cada dos segundos refresca las posiciones de todos los clientes.

El protocolo que sigue esta conexión es la siguiente:

1. Apertura del servidor
2. Conexión del cliente
3. Esperar confirmación
4. Mandar acciones al servidor
5. Si la acción es broadcast, reenvía la acción a los demás usuarios, si es un mensaje de chat, se le envía solo al cliente asociado.
6. Vuelta a 4

3.5.4.2 Entrada estándar

Se basa en el modelo del prototipo 2, pero se ha añadido la posibilidad de recoger eventos del ratón.

Con el ratón podremos elegir el modo en el que empezamos o acceder a la configuración del prototipo.

. Además también le sirve al alumno para poder marcar en el mapa la zona a la que quiere ir, y al guía le ayuda a dirigir al alumno colocando las flechas en el sitio en el cual se ha recibido el evento del ratón.

3.5.4.3 Extras

Se ha añadido un log, en el cual se graban todos los mensajes que se envían los usuarios y los guías. Con esto se hará un análisis posterior para poder ayudar a crear un lenguaje de reconocimiento de peticiones, a través del lenguaje natural.

3.5.5 Versión final

Está basada completamente en el prototipo 3, pero se ha optimizado el código para que su fluidez sea mayor. Además se han incluido elementos para hacer más vistosa la aplicación.

Las teclas de movimiento se han limitado a las teclas de dirección, para que no haya conflicto a la hora de escribir en el chat de comunicación.

Inicialmente se podrá configurar en el menú de configuración, el tipo de sombra, detalles de gráficos, sonido y tipo de conexión (TCP/UDP). Estas acciones estaban anteriormente asociadas al teclado, pero en este prototipo se ha eliminado esa asociatividad.

El mapa se redimensiona pulsando la tecla de función 1, de modo que ocupe toda o que se complemente con la visión del escenario.

Como mejoras graficas, se ha añadido un suelo de hierba sobre el que reposa la facultad, además en cualquier momento puede ponerse a llover, e incluso a nevar simulando así condiciones atmosféricas reales.

También se ha incluido el fuego. En el edificio en diversas zonas aparecerán fuegos, los cuales producen humo, haciendo que la visión del usuario empeore.

Estas dos mejoras graficas se introducen gracias al sistema de partículas que puede implementar Ogre3D (Anexo II: Partículas).

Se puede observar como el fuego hace su aparición en la facultad

3.6 Resumen del capitulo

Tras haber hecho un análisis de los requisitos necesarios para la aplicación, se creó una arquitectura clara y acorde con estos requisitos.

El crear diferentes prototipos ayudó a alcanzar fases estables en el desarrollo de la aplicación, puesto que cada prototipo era una versión estable, y completamente funcional.

La aplicación final queda bastante vistosa gracias también a la inclusión final de partículas, que aportan unos efectos muy llamativos y vistosos.

Capítulo 4:
Discusión

4 Discusión

El proyecto que se ha descrito en este documento ayudara en el futuro a mejorar los actuales protocolos de evacuación que existen, gracias al estudio de las acciones que realizan los usuarios en caso de evacuación por cualquier tipo de emergencia. También ayudara a crear sistemas inteligentes que guíen en tiempo real a las personas que conozcan los protocolos de evacuación. Además el estudio que se puede realizar de las acciones de las personas se puede extrapolar a otros ámbitos que no sean solo el de evacuación. Así de este modo se podría crear un entorno virtual cuyo modelo fuese un museo, y se estudiase los movimientos de los visitantes, y las reacciones antes diferentes cuadros, para que de este modo tras un estudio de las reacciones se pudiese reformar el museo, o crear un guía que ayudase a recorrer el museo de forma más "eficiente" para el visitante. Para realizar este cambio lo único que habría que hacer en el actual proyecto es cambiar el modelo de edificio, y a la hora de estudiar los datos, hacerlo de la manera que mejo convenga al museo en este caso, o al sistema que se quisiese "auditar".

Actualmente se realizan periódicamente simulacros de evacuaciones, en los cuales se aplican los correspondientes protocolos. Para realizar estas evacuaciones hay que movilizar a todo el personal, rompiendo así el ritmo del trabajo, distrayendo por lo tanto al personal. Además al ser un simulacro la tensión que se produce es mínima, la gente no se lo toma en serio, y podrían ir por donde quisiesen. De este modo es imposible controlar al personal, y es imposible saber si el protocolo lo realiza todo el personal correctamente. Por estos motivos se idea y diseño AVANTI (López Mañas, Moreno Nacarino, & Plá Herrero, 2009-2010). Este proyecto aporta una plataforma intuitiva pero completa y potente para usar durante los ensayos de evacuación. Combinando posicionamiento Wifi donde el GPS no llega, predicción de movimiento gracias a los sensores que proporcionan los terminales Android y Realidad Aumentada, para poder contribuir a una experiencia más visual y estimuladora, AVANTI intenta contribuir a que los ensayos de incendio puedan ser informatizados. Estudiantes e instructores puedan analizar fácilmente fallos en los protocolos a seguir, y que llegada la situación de un incendio los ensayos puedan cumplir su objetivo: ser eficaces, y evitar que ninguna persona sufra daños personales.

Si bien es cierto que de este modo se podría estudiar las acciones que realizan las personas durante los protocolos de evacuación, hay que seguir movilizando a todo el personal, rompiendo el ritmo de su trabajo, por ello este proyecto ofrece una alternativa a esa problemática. Un usuario podrá conectarse a través de su terminal a la aplicación, en la cual se desataría una emergencia y habría que poner en práctica los protocolos de evacuación correspondientes. Al igual que con el proyecto AVANTI, también se podrán recoger los datos para luego hacer un estudio de ellos. El usuario podrá elegir cuando le viene mejor realizar los simulacros, de manera que no le impida continuar con su trabajo.

Otro factor a tener en cuenta en las personas que están siendo evacuadas y que no se tiene en cuenta en el proyecto AVANTI es el nivel de estrés que experimentan. Una situación de riesgo provoca en cada persona una reacción diferente y esto no se puede controlar de una forma trivial. Sin embargo, mediante el uso de la aplicación y recreando situaciones de estrés, como la que se comenta a continuación, se podrían obtener resultados fidedignos.

Una posible primera solución hasta que seamos capaces de tener en cuenta esos factores de presión y de estrés de una forma virtual, sería la de someter a los usuarios de nuestra aplicación a situaciones de presión y estrés controlado. Por ejemplo, el hecho de estar privado de respiración se traduce en una respuesta menor a las indicaciones dadas, en los experimentos propuestos se podría hacer que los usuarios realizaran otras tareas mientras intentan seguir las indicaciones que les da la máquina, por ejemplo, teniendo que resolver pequeños ejercicios matemáticos mientras a la vez reciben las indicaciones que deben seguir a lo largo del edificio, para crear la situación de estrés que se puede generar en una situación de peligro .Aunque nunca será posible llegar a tal grado de estrés sin poner en peligro la integridad de los usuarios, estas pruebas pueden darnos una idea de los tiempos de reacción de la gente y del nivel de concentración que pueden llegar a tener, haciendo necesario, a lo mejor, tener que repetir las indicaciones más de una vez, cosa que en un primer momento puedes no llegar a tener en cuenta.

Otra alternativa más agresiva seria usar estimulación eléctrica, conectando al usuario una serie de electrodos, que produjesen pequeñas descargas cuando el protocolo no se realizase correctamente.

La empresa SIN (Seguridad Integral del Norte) (Norte) ofrece la creación e implantación de un plan de Autoprotección. La manera que tienen de simular las evacuaciones es mediante un video 3D en el que se enseña la manera de hacer una evacuación de la forma correcta. Este método es demasiado impersonal debido a que los usuarios por mucha atención que quieran prestar pueden distraerse en cualquier momento, y de esto modo no conocer las medidas completas que hay que tomar en caso de evacuación, a diferencia del proyecto que se ha explicado durante todo el documento en el cual el usuario debe realizar por sus propios medio el protocolo de evacuación, siendo así mucho mas interactivo, y pudiendo rectificar al usuario en tiempo real si no lo realiza correctamente.

En el proyecto realizado por Víctor Romero Pérez para la universidad Carlos III, Creación de un entorno 3D para la simulación de tráfico urbano, se da la opción de crear un entorno para generar simulacros, en este caso como su nombre indica de tráfico urbano. Eso implica que haya que configurar el proyecto cada vez que se use, con su consiguiente gasto de tiempo a la hora de configurarlo, y la necesidad de tener a personal que tenga los conocimientos para configurarlo. El simulado de entorno 3D creado en este proyecto, no necesita ser configurado previamente, el usuario selecciona el simulacro que quiera realizar, y automáticamente se ejecutara y podrá empezar a realizarlo.

Con los datos recogidos de las simulaciones, se podrá ayudar a crear un sistema de ayuda, con el que dar consejos útiles a personas que se encuentren en situación de peligro previamente estudiadas. Estos datos son recogidos en tiempo real, gracias a las conversaciones que se realizan entre los guías y los alumnos en el caso de este proyecto. Es recomendable realizar las pruebas para recoger datos con usuarios que no conozcan el recinto, debido a que si lo conocen, pueden no solicitar información, y las pruebas no servirían.

4.1 Objetivos alcanzados

El proyecto desarrollado es específico para la recreación de situaciones de fuego real en edificios cerrados. Gracias a los datos obtenidos mediante el uso de la aplicación, pueden crearse nuevos protocolos de evacuación o mejorar los existentes. Otro factor positivo es la considerable reducción de coste que supone el poder evaluar

una y otra vez diferentes tipos de incendios, así como diferentes formas de solucionarlos, sin tener que recrearlas en la vida real. En la recreación de una situación de riesgo provocada por un fuego en un edificio cerrado, se necesitaría:

- Personal profesional que actúe en este tipo de contingencias, es decir, policías, bomberos, sanitarios, etc.
- Personas que se hagan pasar por los accidentados, que pueden ser de los propios cuerpos profesionales o personas ajenas a ellos.
- Un edificio reservado exclusivamente para dicha recreación.
- Material para simular el incendio, véase: camiones de bomberos, ambulancias, coches patrulla, el agua que se utilice para sofocar el fuego, etc.
- Una preparación previa de organización y planificación de la situación a simular.
- Personal que vigile la simulación y recoja los datos y las impresiones obtenidas para un posterior análisis.

Todo este despliegue de medios es necesario para recrear una única situación, una única vez. Se puede valorar el esfuerzo necesario y el gran coste que esto representa. Sin embargo, con el uso de la aplicación desarrollada, cada situación a recrear puede realizarse las veces que se quiera. Además, se pueden combinar todas las posibilidades imaginables, aumentando el número de usuarios, teniendo que rescatar a una persona aislada, probando diferentes vías de escape. Incluso se pueden simular los efectos provocados por la mala gestión de una situación de peligro, cosa que en un entorno real sería inviable realizar, sin poner en riesgo la integridad física de ningún participante.

El coste necesario para realizar dichos simulacros mediante la aplicación es mucho menor. La infraestructura necesaria radica en un grupo de usuarios que participen en dicha simulación, dependiendo del número de accidentados que queramos incluir, así como de los terminales necesarios para dichos usuarios.

La manera de poder aumentar el número de usuarios se da gracias a que la aplicación está pensada para que sea multiusuario, de este modo cada cliente que se conecte, creara un nuevo usuario que se podrá visualizar en el entorno. Gracias a la inclusión del sistema multiusuario se podrán crear simulacros conjuntos entre varios usuarios, recreando una situación aun mas real así. Esto también posibilita que dos guías estén dirigiendo a un mismo alumno de manera que pueden proporcionarle información diferente para cumplir los protocolos de evacuación, que por una parte puede crear confusión al usuario que está siendo guiado, pero puede a la vez ayudarle en caso de que las opciones que dé el otro guía no sean muy adecuadas.

La aplicación resultante pese a que puede tener una gran cantidad de mejoras, ha resultado ser muy estable y completa para el caso concreto de evacuación a causa de un incendio. El usuario puede ver los fuegos que hay en el recinto, y de este modo dirigirse a la salida evitando estos de un modo más real que en un simulacro real, en el que la posibilidad de incluir fuego sería muy peligrosa. Además se pueden incluir personajes con inteligencia artificial, a los cuales halla que rescatar, y todo esto sin peligro real para las personas. Incluye la capacidad de crear simulacros nocturnos lo que conlleva una menor visibilidad, de manera que mucho más cómoda para los usuarios, que se ahorrarían el tener que ir a horas no deseables para realizar simulacros nocturnos.

4.2 Objetivos futuros

El proyecto desarrollado no es el ideal de aplicación y tiene partes donde todavía la fase de desarrollo tiene alguna carga importante, aunque el poder recrear muchas situaciones de peligro, originadas por el fuego, sin poner la vida de nadie en peligro, es un gran paso.

La aplicación desarrollada actualmente solo cubre situaciones de peligro provocadas por incendios en edificios cerrados. Futuros pasos pueden llevar a desarrollar diferentes ampliaciones que cubran otras situaciones de peligro, como por ejemplo: derrumbamientos de edificios (por terremotos u otras contingencias), inundaciones, rescates en accidentes de tráfico, etc.

Actualmente cualquier evacuación que se realizase en un entorno cerrado podría implementarse sin prácticamente cambios, solo habría que cambiar el modelo de edificio y añadir elementos al mapa que simulasen por ejemplo agua en el caso de inundación, haciendo además que el terreno fuese resbaladizo modificando para ello la física del mundo. Se podría simular un terremoto moviendo para ello el escenario también mediante física.

Podemos sacar en conclusión que modificando el tipo de física, y el modelo de edificio bastaría para simular un numero casi ilimitado de evacuaciones.

Como punto para futuras ampliaciones del proyecto actual en el que se simula un incendio sería interesante el poder contemplar todos los aspectos originados por un fuego real, como por ejemplo, los problemas respiratorios provocados por el humo y su efecto en la concentración y capacidad de reacción de las personas, así como el calor y sus consecuencias en los afectados. Se podría lograr un efecto de mareo haciendo que el usuario no viera la pantalla correctamente, y a la vez que las teclas no respondiesen como el usuario esperase.

Se podría hacer que la aplicación fuese evolucionando por sí misma en el sistema de autoayuda y guía automática aprendiendo de la información que se ha obtenido mediante el uso de la aplicación desarrollada, tratándola de una forma adecuada.

Los datos obtenidos ayudarían mucho a ponerse en la piel de las personas en situaciones de peligro y obtener un muy buen punto de partida para establecer nuevos protocolos de emergencia o mejorar los actuales.

4.3 Otros usos de la aplicación

El objetivo principal del proyecto desarrollado es el de poder guiar a una o varias personas a través de un edificio durante una situación de peligro provocada por un fuego para que de este modo exista una comunicación entre el guía y la persona a la que guía, la cual será después procesada y gracias a ella se podrá crea un sistema automático de guías y ayuda para evacuaciones.

Otro uso sería el de ser un mero guía que te indique cómo llegar a un sitio concreto. El proyecto se desarrolla sobre el modelado de la facultad de Informática de la universidad Complutense de Madrid. Supongamos a una persona ajena a la facultad que viene a la misma para realizar cualquier tipo de actividad, como por ejemplo: asistir a un congreso que se desarrolle en la facultad, buscar un libro o una publicación de la biblioteca, etc. El proyecto podría proporcionarle la ruta hacia la sala de conferencias o a la biblioteca, respectivamente. Este uso en la facultad puede no resultar muy eficaz ya

que la superficie de la misma no es muy extensa y los sitios se encuentran relativamente cercanos y con facilidad, pero extrapolándolo al modelado de edificios más grandes y con distribuciones más complicadas podría ser una ayuda muy útil.

Un ejemplo de edificio que puede beneficiarse de este uso sería un museo. Una vez hecho el modelado del museo se podría indicar al usuario la ruta para poder ver un cuadro en particular. Todos los museos cuentan con una exposición fija que permanece siempre en el mismo lugar, pero la mayoría cuentan con exposiciones itinerantes, y la ubicación de los cuadros es cambiante. Una vez que se tiene el modelado, ubicar el cuadro en el mismo es algo muy sencillo que no conlleva un gran esfuerzo. Por lo tanto la renovación de las ubicaciones sería totalmente actual y sería una ayuda a la hora de encontrar el cuadro deseado de una forma rápida.

Otro uso interesante sería el de poder usar la aplicación desarrollada para poder guiar a personas con deficiencias visuales. Mediante la interacción con su dispositivo móvil y el posicionamiento Wifi para edificios cerrados, la persona invidente podría seleccionar el lugar al que desea ir y mediante las instrucciones generadas automáticamente por la aplicación poder llegar al lugar adecuado. Además, la aplicación puede avisarle de las peculiaridades del edificio, como pueden ser puertas que se encuentren cerradas por motivos de seguridad, elementos del mobiliario (bancos , mesas, sillas, etc.) y así proporcionarle información útil para que la acción de llegar al sitio indicado sea lo más fácil posible.

4.4 BerliOS y el uso de software libre

Para el desarrollo del proyecto se ha utilizado el servidor de BerliOS (Berlín Open Source), al tratarse de código abierto, el proyecto se encuentra a disposición de todo aquél que quiera consultarlo. El hecho de que otras personas puedan beneficiarse gracias al proyecto desarrollado para empresas personales es algo que crea un sentimiento de comunidad, que el uso de software libre hace patente.

Sin pretender desmerecer el uso de software de propietario, se van a exponer a continuación las ventajas tenidas en cuenta para elegir el uso de software de libre distribución:

- La principal y más importante es el coste. El uso de software libre por parte de PYMES permite reducir el coste derivados de la adquisición de licencias procedentes de software de propietario. Si es cierto que el software de propietario se hace necesario para aplicaciones específicas y de baja demanda.
- La libre distribución y la portabilidad que proporciona el software libre es otra característica que le hace muy atractivo. La mayoría de las personas cuentan actualmente con más de un terminal, ya sean ordenadores de mesas, portátiles o netbooks. La posibilidad de poder instalar dicho software en cualquier terminal sin tener problemas de licencias es muy ventajoso a la hora del desarrollo de proyectos en los que se deben combinar el uso de los distintos terminales.
- El acceso al código fuente permite desarrollar nuevas aplicaciones sin tener que empezarlas desde cero, se puede reutilizar código ya escrito y minimizar el tiempo de desarrollo de nuevos proyectos.
- Se fomenta la libre competencia al basarse en servicios y no en licencias. Uno de los modelos más importante que genera el software libre es el de

servicios de atención al cliente. Esto permite que las diferentes empresas que ofrecen el servicio compitan en igualdad de condiciones al no poseer la propiedad del producto. Todo esto repercute en una mejor atención al cliente.

- La compatibilidad a largo plazo también es muy interesante, esto es una característica innata al software libre que se la proporciona el propio software de propietario por sus características. Llegado al número máximo de ventas no interesa que los usuarios sigan con este software con lo que se desarrollan nuevos para poder obtener beneficios de nuevo. El ejemplo lo encontramos en las diferentes versiones de Windows.
- Una de las ventajas más notables es la de la creación de formatos estándar. Los formatos estándares afectan a todos los niveles. Un ejemplo de esto son los documentos emitidos por las administraciones públicas con formatos y versiones diferentes que lo único que producen son retrasos y dificultades a la hora de acceder a información. Algunas administraciones Europeas están dando los primeros pasos hacia formatos abiertos , como por ejemplo: ODF (Open Document Format).
- Sistemas más seguros, esto se debe al hecho de que al tratarse de código fuente de libre distribución, las auditorías de los programas por parte de empresas de seguridad informática es total y el uso de trampas como pueden ser las puertas es ilógico ya que perjudica a la comunidad que desarrolla la aplicación.
- Otra ventaja derivada del código fuente libre es la de poder permitir una corrección de fallos más rápida y más eficiente. El sentimiento de comunidad generado por el software libre hace que haya muchos ojos observando el código y se ha demostrado que los fallos de seguridad que puedan tener los programas son rápidamente solucionados. Esto se puede reconocer en la enciclopedia libre Wikipedia. Si existe un error en la definición de algún término, el tiempo de corrección es muy rápido debido al número de personas que participan de él.

Es por todo lo anteriormente comentado, la razón del uso de software libre frente al software de propietario en el desarrollo del proyecto. Habiéndose beneficiado todo el proceso de las ventajas del mismo.

4.5 Herramientas

En este apartado se van a comentar algunas de las herramientas utilizadas en el proceso de desarrollo del proyecto y las características o ventajas que nos han llevado a su utilización.

Una de estas herramientas es Git. Se trata de un sistema de control de versiones distribuido, de código libre. Algunas de las ventajas que se han tenido en cuanta para su uso frente a otros se detallan a continuación:

- La diferencia principal de GIT frente a otros sistemas de control de versiones como puede ser CVS o SVN radica en que estos últimos cuentan con un repositorio central con el cual se sincronizan todos los integrantes del proyecto. Sin embargo, en Git la idea de un repositorio central no existe, y el repositorio está distribuido entre todos los integrantes del proyecto. Esto quiere decir que cada uno de los participantes cuenta en local con un histórico, etiquetas, ramas, etc. La

principal ventaja es que no tenemos que estar conectados a la red para hacer cualquier operación contra el repositorio, con lo que se consigue una mayor rapidez en el trabajo y menos dependencias.

- Como se ha comentado es útil para conexiones lentas o si no se dispone de una conexión continúa.
- Es el más adecuado para equipos grandes, con poca comunicación o con localizaciones geográficas diferentes.
- Se puede incluir a personas ajenas al proyecto que queremos que participen en cierta fase del mismo sin necesidad de que pertenezcan a la organización y sin necesidad de permisos. En estos casos la inclusión del código se lleva a cabo mediante parches.

Otra herramienta destacada ha sido Eclipse, y más concretamente el plugins denominado PyDev, que proporciona las herramientas para el desarrollo de proyectos bajo el lenguaje Python. Gracias a ello, la depuración y las pruebas del código son mucho más eficientes que en el entorno proporcionado al instalar Python. Enfrentarse a un lenguaje nuevo como es Python resulta menos agresivo si se cuenta con una interfaz conocida, como Eclipse, en la cual ya se ha trabajado bajo otros lenguajes como Java o C++, entre otros.

Para la creación del personaje o avatar que se utiliza en el proyecto se ha utilizado el programa Blender. Se trata de un programa informático multiplataforma, que es específico para el modelado, animación y creación de gráficos. Es un programa de libre distribución y una de las ventajas más atractiva es el tamaño reducido comparado con otros paquetes de 3D, aunque también depende del sistema operativo utilizado.

Capítulo 5:
Conclusiones y trabajo futuro

5 Conclusiones y trabajo futuro

5.1 Resumen del proyecto

Este proyecto es una implementación en 3D de un entorno virtual que servirá para simular incendios en la Facultad de Informática de la Universidad Complutense de Madrid. Es un entorno multiusuario cliente – servidor bajo el protocolo de red TCP. Es un proyecto en código libre disponible en el servidor Berlios bajo el nombre de Trumpet.

Existen dos roles en el sistema que el usuario puede asumir: Alumno y guía, donde los primeros son guiados por los segundos a través de los pasillos de la facultad hacia la salida de emergencia más segura.

El entorno virtual es multiusuario y permite a los usuarios moverse por el mismo entorno virtual, viéndose unos a otros y comunicándose a través de un chat.

5.2 Relación con el Proyecto del Plan Nacional

El proyecto está encuadrado dentro del "Plan Nacional de I+D+i del Ministerio de Ciencia e Innovación", su nombre es "Navegación basada en Ontologías mediante Verbalización de mensajes de Ayuda (NOVA). Es un proyecto a tres años, desde 2010 a 2012 con el grupo NIL (Natural Interaction base on Language) dirigido por Pablo Gervás.

Por la tanto, este proyecto una pieza de un puzle, y como tal está encajado con las piezas que ya existían, a la vez que permite encajar a las piezas que vendrán. El rol de este proyecto en NOVA es el desarrollo del entorno virtual en el que poner en práctica las simulaciones de los sistemas inteligentes automáticos de guía y gestión de ayuda a usuarios que se realicen.

5.3 Beneficios e inconvenientes de las soluciones dadas

Gracias a este proyecto se podrán mejorar los protocolos de evacuación actuales, gracias al estudio que se realizaran mediante estas simulaciones. Además los simulacros se podrán realizar en cualquier momento, de manera que no será necesario realizar un plan previo para movilizar a todo el personal. De este modo es mucho más cómodo para una persona realizar el simulacro de manera que no se rompa su ritmo de trabajo. Además se podrán simular múltiples situaciones, sin ningún coste adicional, pudiéndose realizar el número de veces que sean necesarias y sin correr ningún riesgo.

La gran desventaja es no ser una situación real, de manera que la intensidad con que se realiza no es igual que si fuera una emergencia real, de manera que los datos obtenidos hay que interpretarlos con esta idea siempre presente.

5.4 Satisfacción de objetivos

Había establecidos una serie de requisitos que debían cumplirse. Se decidió agrupar estos objetivos en diferentes prototipos e imponer plazos para la terminación de cada uno, asegurando la finalización de todos ellos. A continuación se enuncian y se explican brevemente estos objetivos indicando a su vez como fueron resueltos

5.4.1 Resumen de los objetivos propuestos

1. Desarrollo de un entorno virtual libre en 3d en el que un personaje pudiera ser movido por el usuario en cualquier dirección

2. Integrar un motor de físicas de forma que los personajes reaccionen a las colisiones con los diversos objetos que componen el escenario así como el resto de personajes.

3. Incluir un repertorio de cámaras, luces y sombras, que pudieran ser seleccionadas de forma sencilla por el usuario.

4. El entorno virtual debía ser multiusuario, integrando una red de forma que diferentes usuarios pudieran acceder al entorno virtual de forma simultánea, transmitiendo sus movimientos de unos a otros.

5. Añadir funcionalidad a la red para que los usuarios pudieran interaccionar entre ellos transmitiendo mensajes a través de ella utilizando un chat.

6. Los usuarios deben de poder elegir entre dos roles diferentes, alumno o guía, donde los primeros son guiados por los segundos hacia las salidas de emergencia más seguras en la simulación de un incendio.

5.4.2 Cumplimiento de objetivos

A continuación se explica el modo en que se han resuelto cada uno de los objetivos citados anteriormente.

1. Se creó un escenario con varias habitaciones y puertas, así como diferentes objetos dispersos, se añadió un personaje y se incluyeron un detector de colisiones y un motor de sonido.

2. Se estudiaron diferentes motores de físicas eligiendo finalmente OgreODE, el motor reaccionaba a las colisiones entre personajes y con los diferentes objetos situados en el escenario.

3. Se han desarrollado dos tipos diferentes de vistas, la primera es una cámara libre en primera persona y la segunda se trata de una cámara acoplada a un personaje que sigue sus movimientos. También hay diferentes tipos de luces, natural, con las luces apagadas y una luz roja intermitente que simula luces de emergencia.

4. La red está implementada mediante el modelo cliente-servidor, usando la librería de Ogre RakNet. En un primer momento se utilizó el protocolo de red UDP, y posteriormente se cambió a TCP debido a la pérdida de paquetes que se producía con el primer protocolo.

5. Haciendo uso de la red implementada se incluyó la posibilidad de que los personajes se comunicaran entre ellos, usando CEGUI para incluir en la interfaz gráfica una ventana donde los usuarios pueden escribir mensajes que posteriormente son transmitidos por la red.

6. Se establecieron los dos tipos de roles de forma que cada guía cuando accede al entorno virtual puede elegir entre todos los alumnos para elegir a quien guiar, selecciona uno en ese momento se abre un canal de comunicación por chat entre los dos usuarios. De esta forma, cada alumno puede tener uno o más guías, pero cada guía solo puede guiar a un alumno.

5.4.3 Balance general

El desarrollo de un entorno virtual es un trabajo que conlleva muchas dificultades. Muchas de ellas no están previstas en un primer momento y las soluciones que se deben aportar para resolverlas no son triviales. A pesar de todo, y tras haber podido resolver de forma más o menos adecuada todas las complicaciones encontradas, se ha conseguido implementar el 100% de la funcionalidad pedida para el proyecto y podemos considerar que el balance general es muy positivo.

5.4.4 Ampliaciones del proyecto

Hay multitud de posibilidades que permitirían a este proyecto mejorar en cuanto a funcionalidad, usabilidad y estética. Algunas de las ideas que incluimos a continuación son proyectos de Sistemas Iinformáticos de otros grupos que podrían ser integrados en este sistema, e ideas que surgieron durante el desarrollo del proyecto que no pudieron ser incluidas por diferentes razones.

El entorno virtual de simulación actual se limita a la planta tercera de la facultad de informática. Se disponía del modelo completo del edificio, pero se decidió reducirlo por razones de eficiencia. Una posible ampliación sería tratar de aligerar el modelo del edificio disminuyendo el número de mayas que lo componen, o encontrando la manera de cargar dinámicamente aquellas que sean estrictamente necesarias.

Desde el departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la UCM se dirigió un proyecto consistente en un generador de personajes inteligentes. Este módulo podría acoplarse al entorno virtual de forma que se pudiera simular una multitud que huye hacia la salida de emergencia, que producen embotellamientos, sufren caídas etc.

También se están realizando estudios sobre la interpretación del lenguaje natural por un sistema. Las conclusiones obtenidas podrían ser aplicadas para que la interacción entre los usuarios del entorno virtual no tuviera que realizarse necesariamente mediante un teclado y pudieran ser, por ejemplo, hablando a través de un dispositivo.

El rol del guía podría ser sustituido por un agente dotado de inteligencia artificial que en caso de incendio fuera capaz de tomar decisiones en tiempo real, guiando a los usuarios hacia las salidas de emergencia mediante una megafonía o utilizando paneles luminosos.

Podría también hacerse uso de técnicas de Realidad Aumentada para integrar el entorno virtual desarrollado con el entorno real. Se podría desarrollar una aplicación para Smartphone que incluyera el entorno virtual y permitiera conocer la localización de otros usuarios y del fuego, a la vez que se reciben indicaciones de los guías mirando la pantalla del dispositivo.

5.4.5 Aplicaciones del proyecto

A parte de las aplicaciones para las que fue concebido el proyecto puede también usarse para otros medios sin la necesidad de realizar demasiados cambios sobre la implementación actual.

Puede desarrollarse algún videojuego que transcurra en la facultad informática, tipo shot´em up o de estrategia, o bien, solo con incluir en el entorno virtual las diferentes piezas del museo García Santesmases podrían realizarse visitas virtuales a través de internet.

Desarrollo de ferias virtuales. Puede emplearse el espacio del entorno virtual para que distintas empresas y organismos sitúen sus stand para ofrecer a los alumnos ofertas de prácticas y contratos laborales, o informen sobre diferentes programas de estudios.

Se podría ejecutar el entorno virtual en un servidor web para poder visitar virtualmente el edificio desde otra localización. El entorno virtual trataría de asemejarse lo máximo posible a la actividad real que estuviera viviendo en ese momento, haciendo posible, por ejemplo, asistir por videoconferencia o simulación 3D a las lecciones y conferencias que se estén impartiendo, consultar los tablones de anuncios, acudir a tutorías, organizar reuniones etc. Pudiendo llegar, en un momento dado, a volcar toda la información de la página web en el entorno virtual.

Bibliografía

3, U. E. (s.f.). *http://www.unrealengine.com/*.

Abella, A. (2004). *Libro Blanco sobre el software Libre en España.*

Arturo F. Montagu, C. C. (s.f.). *Urbamedia, desarrollo de una base de datos de fragmentos urbanos de ciudades argentinas y latinoamericanas utilizando tecnología VRML*. Universidad de Buenos Aires.

Bonnin, J. (s.f.). *¿Que es la realidad Aumentada? http://outernet.tk/.*

Brunet, P. (2002). *Realidad virtual, la verdad en tres dimensiones.*

Burgess, A. (2010). *Getting Good with Git.* Rockable.

Chronister, J. (2006). *Blender Basics- 2nd Edition.*

Elvira San Millán Fernández, M. L. (2008). *Social Media Marketing, redes sociales y metaversos.*

Epsilon. (s.f.). *Medisolae 3D. www.medisolae-3d.eu.*

Horsnell, M. (Abril de 2005). Sony takes 3-D cinema directly to the brain. *The Sunday Times* .

Jose R. Hilera, S. O. (s.f.). *Aplicación de la Realidad Virtual en la enseñanza a través de Internet.* Universidad de Alcalá de Henares.

Junker, G. (2006). *Pro Ogre 3D Programming.*

Kerger, F. (2010). *OGRE 3D 1.7 Beginner's Guide*.

León, C., de la Puente, S., Dionne, D., & Gervás, P. (2009). A Model for Human Readable Instruction Generation Using Level-Based Discourse Planning and Dynamic Inference of Attributes Disambiguation.

López Mañas, E., Moreno Nacarino, F. J., & Plá Herrero, J. (2009-2010). *Proyecto Avanti.*

Miguel Lozano, C. C. (s.f.). *Entornos virtuales 3D clásicos e inteligentes: hacia un marco de simulación para aplicaciones gráficas 3D interactivas.* Universidad de Valencia.

Nintendo. (2011). *www.nintendo.es.*

Norte, S. I. Seguridad Integral del Norte., (pág. http://www.segurintenorte.es/).

Ogre. (s.f.). *www.ogre3d.org*.

OpenAl. (s.f.). *http://connect.creativelabs.com/openal/default.aspx.*

Pérez, V. R. (2009). *Creación de un entorno 3D para la simulacion de tráfico urbano.*

Pérez., V. R. (2009). *CREACIÓN DE UN ENTORNO 3D PARA LA SIMULACIÓN DE TRÁFICO URBANO.* Madrid.

Python. (s.f.). *www.python.org*.

Rage. (s.f.). *http://www.rage.com/*.

RV. (s.f.). *www.realidadvirtual.com.*

Silva, A. P. (2007). *Niveles de tecnología de realidad virtual http://www.monografias.com/trabajos53/realidad-virtual/realidad-virtual2.shtml#nivel.*

SourceEngine. (s.f.). *http://source.valvesoftware.com/.*

Swicegood, T. (2009). *Pragmatic Version Control Using Git.*

Tecayehuatl, E. (2009). *Cyberwalk: Caminadora omnidireccional para mundos virtuales.* Puebla (México).

Torrente Vigil, F. J., Cañizal Alzola, G., & del Blanco Aguado, Á. (2007/2008). *Entorno de Autoría Para la Creación de Aventuras 3D Educativas en Entornos Virtuales de enseñanza.*

UnrealEngine3. (s.f.). *http://www.unrealengine.com/.*

Anexo I: Manual de uso

A continuación se detallara los pasos a seguir para poder ejecutar la aplicación.

En primer lugar es necesario que exista un servidor al cual conectarse. Si no se dispone de uno, el propi programa cuenta con dos servidos uno TCP y otro UDP.

Tras inicializar el servidor ya se puede ejecutar el programa, mostrando por pantalla las diferentes opciones de inicio

Pantalla de selección de modo, o de configuración

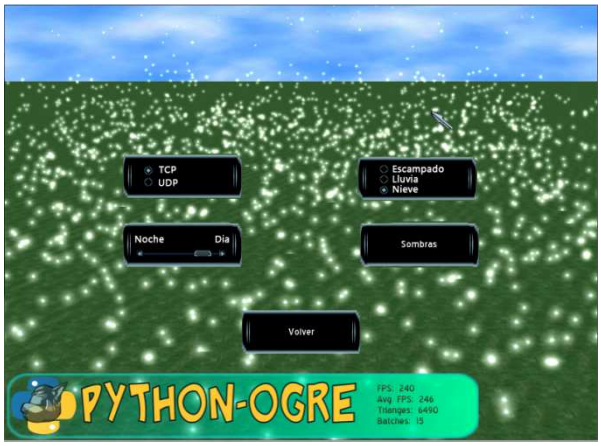
Lo primero será configurar el simulacro mediante el menú de opciones

Se muestran las diferentes configuraciones posibles

En este menú se podrá elegir al servidor que se desea conectar, TCP o UDP.

El usuario podrá realizar el simulacro bajo diferentes condiciones meteorológicas, como pueden ser lluvia, nieve o cielo despejado.

El usuario ha elegido lluvia



En esta ocasión la elección fue que nevara

Otra posibilidad es elegir la hora del día a la que se realiza, modificando de este modo la cantidad de luz solar.

La simulación se decide simular por mientras anochece

El ultimo aspecto configurable es el tipo de sombras que se decide usar, conllevando esta elección una mejora en el nivel grafico pero bajando el rendimiento de la aplicación, o viceversa.

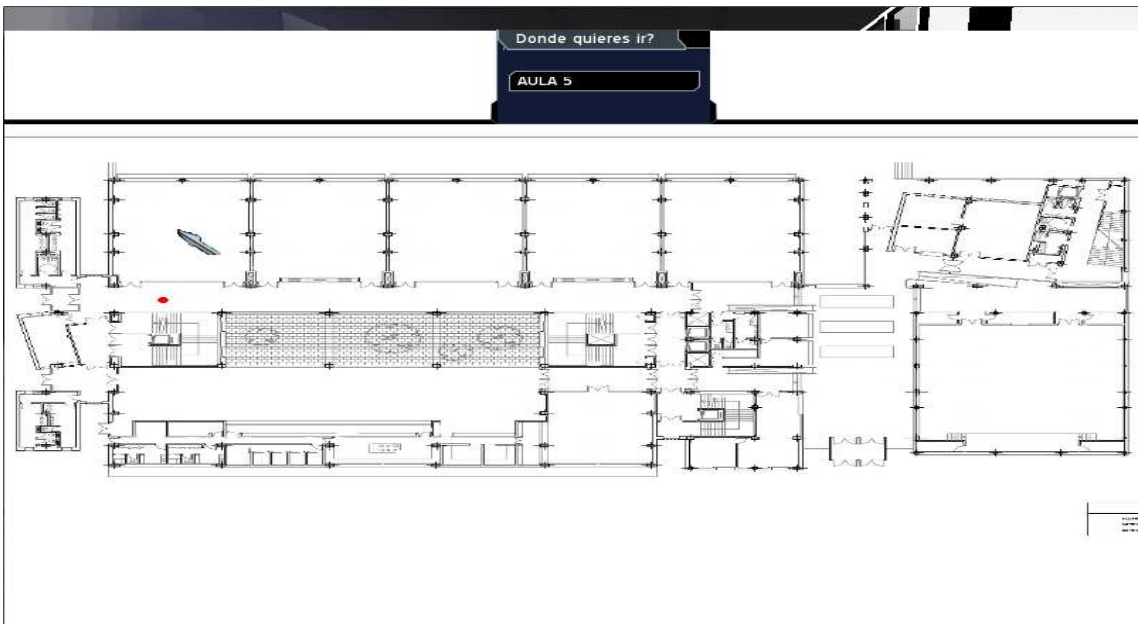
Una vez configurad el proyecto es hora de elegir el tipo de usuario con el que se quiera entrar en la aplicación.

Si la elección es Alumno, se inicializara la interfaz grafica del Alumno, en la cual el usuario podrá moverse libremente a la espera de que un guía le ayude.

La imagen muestra como un nuevo Alumno se ha conectado, y esta la espera de que le guíen. Se puede observar como otro alumno delante está siendo ya guiado

Una vez que el Alumno dispone de un guía, este puede pedirle ayuda a través del chat, o a mediante el mapa del que dispone el alumno.

El mapa se redimensiona pulsando la tecla de función 1. Si se ha pulsado y esta maximizado, el Alumno podrá pinchar sobre él para indicarle al guía donde desea ir.

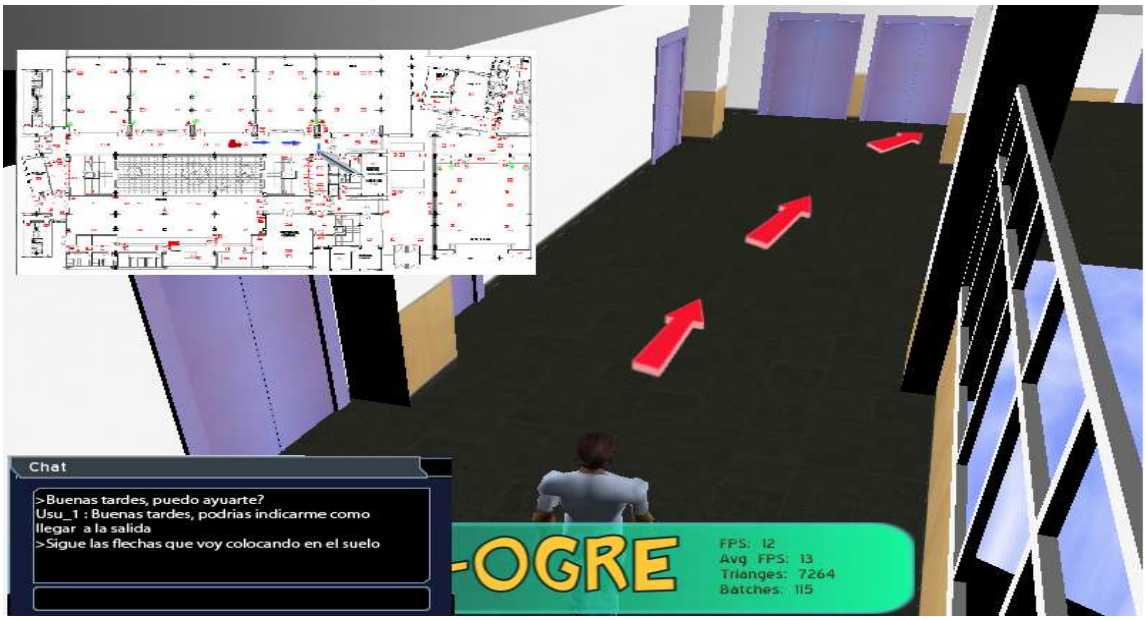


El alumno ha elegido ir al aula 5. Se puede ver en que sitio se encuentra el alumno gracias al punto rojo que indica su posición1

La Otra opción de inicio es entrar a la aplicación en modo Guía. Si se decide por entrar en modo Guía, la interfaz mostrara primero una lista de todos los Alumnos conectados, de los cuales el Guía deberá elegir uno al que ayudar.

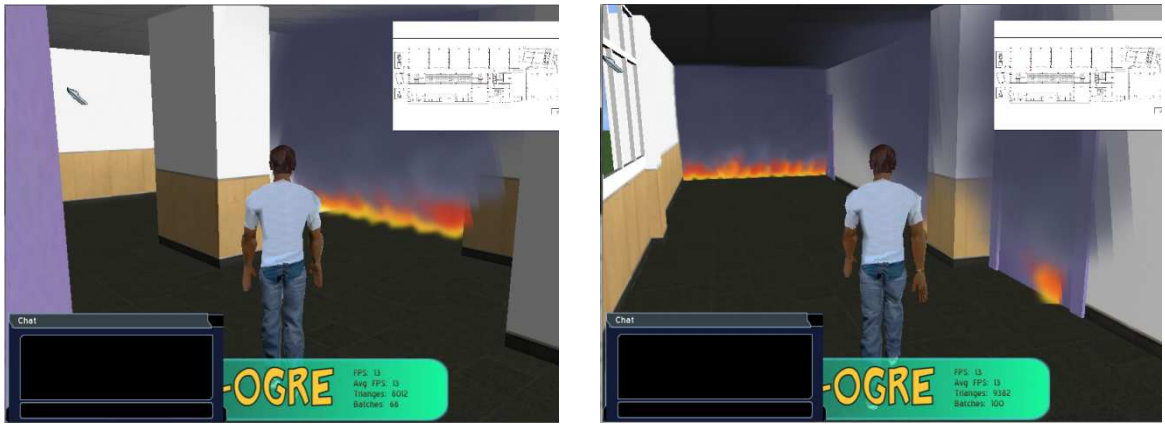
El guía deberá elegir entre uno de los tres Alumnos para poder continuar

Una vez elegido, el Guía podrá ayudar al Alumno a través del chat dándole órdenes, o mediante su mapa, de manera que pulsando sobre él, creara en su interfaz grafica y en la interfaz grafica del correspondiente alumno una flecha que indicara el camino a seguir



Interfaz grafica del Guía que ha indicado al alumno hacia dónde dirigirse mediante flechas indicativas

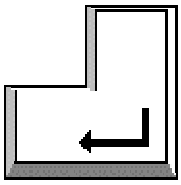
Uno de los elementos que aparece en ambas interfaces, es el fuego. El fuego impide que el alumno pueda seguir si se encuentra con el.



Diferentes lugares, por los cuales el fuego impide el paso del alumno

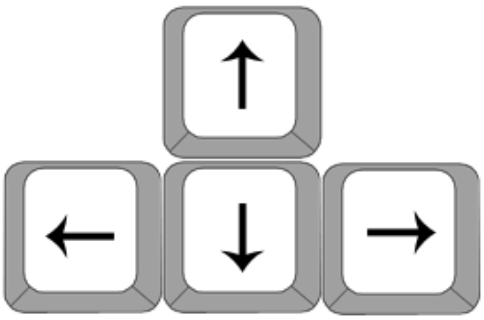
Controles

Teclas para poder comunicarse a través del chat



Envía el mensaje que se ha escrito

Maximiza/minimiza el mapa



Movimiento del personaje

Alumno: Al pinchar en el mapa indica al guía al lugar al que desea ir

Guía: Sitúa en el mapa elementos indicativos para el alumno

Anexo II: .partículas

Las partículas agregan espectacularidad a la aplicación. Son efectos visuales en 2D rápidos de dibujar en pantalla y bonitos a la vista. Se manejan en el código usando un objeto de tipo ParticleSystem*, y se le pide crear un nuevo sistema de partículas al ParticleSystemManager. Las partículas se pueden mostrar, ocultar (con setVisible (true o false), igual que las entidades), pero hay que mencionar que no se verán en pantalla hasta que no sean adjuntas a un Nodo. Las partículas se adjuntan a un nodo al igual que las entidades u otros nodos.

ParticleSys=

sceneMgr.createParticleSystem ("fuego"+String (numero),"particulas/fuego")

/ El primer parámetro de createSystem es el nombre de la partícula, en Ogre debe ser un nombre único. El segundo parámetro es una partícula del fichero de partícula que se tenga. Debe estar incluida en la carpeta de fuentes de Ogre */*

Node.attachObject (particleSys); //Adjuntamos la partícula para que se vea

Solo con eso la partícula ya se muestra adjunta al nodo en pantalla. Las partículas se animan y hacen todo solo, dependiendo de los parámetros que les hayan puesto en su archivo.

Las animaciones de las partículas se guardan en archivos .particle en las carpetas media de Ogre. Allí se crea una partícula con su nombre que puede ser usada luego en el código se ha visto. Si se hace un cambio a este archivo de configuración de la partícula para que se vea distinta, más rápida, de otro color o con otra imagen, no es necesario recompilar el programa para que el cambio haga efecto (son como los materiales de las texturas en ese aspecto).

```
particle_system particulas/fuego
{
        Material
partículas/fuego
        particle_width 55
        particle_height        55
        cull_each      true
        quota          500
        billboard_type point
                sorted
        true

        // Area emitter
        emitter Point
        {
                angle          11
        emission_rate          15
        time_to_live           4
        direction              0 0
        velocity_min           50
        velocity_max                   100
                }

                affectorColourImage
                {
                        image
        smokecolors.png
                }
                affector Rotator
                {
                rotation_range_start   0
                rotation_range_end 360
                rotationspeedrangrt    -60
                rotation_speed_range_en
        d  200
                }

                affectorScaler
                {
                rate    1
                }
```